

# Dr. Dobb's Journal of Software Tools

FOR THE PROFESSIONAL PROGRAMMER

**Handling Large  
Priority Queues**

**TSR Serial Driver**

**Analog-to-Digital  
Conversion**

**Unix Shell Scripts**

**Languages:**

Dynamic Overlays in Turbo Pascal

Handling Queues in C

AI Programming in SCOOPS

User-Defined Functions in BASIC







# Eureka: The Solver™

NOW SHIPPING!

**A**n anyone and everyone who routinely works with equations needs **Eureka: The Solver**

It solves the most complex equations in seconds. Whether you're a scientist, engineer, financial analyst, student, teacher, or some other professional, you need Eureka: The Solver!

Any problem that can be expressed as a linear or non-linear equation can be solved with Eureka. Algebra, Trigonometry and Calculus problems are a snap.

Eureka: The Solver also handles maximization and minimization problems, does plot functions, generates reports, and saves you an incredible amount of time.

**$X + \exp(X) = 10$   
solved instantly instead  
of eventually!**

Imagine you have to "solve for X," where  $X + \exp(X) = 10$ , and you don't have Eureka: The Solver. What you do have is a problem, because it's going to take a lot of time guessing at "X." Maybe your guesses get closer and closer to the right answer, but it's also getting closer and closer to midnight and you're doing it the hard way.

With Eureka: The Solver, there's no guessing, no dancing in the dark—you get the right answer, right now. (PS:  $X = 2.0705799$ , and Eureka solved that one in .4 of a second!)

## How to use Eureka: The Solver

It's easy.

1. Enter your equation into the full-screen editor
2. Select the "Solve" command
3. Look at the answer
4. You're done

You can then tell Eureka to

- Evaluate your solution
- Plot a graph
- Generate a report, then send the output to your printer, disk file or screen
- Or all of the above

## Eureka: The Solver includes

- ✓ A full-screen editor
- ✓ Pull-down menus
- ✓ Context-sensitive Help
- ✓ On-screen calculator
- ✓ Automatic 8087 math co-processor chip support
- ✓ Powerful financial functions
- ✓ Built-in and user-defined math and financial functions
- ✓ Ability to generate reports complete with plots and lists
- ✓ Polynomial finder
- ✓ Inequality solutions

## Some of Eureka's key features

You can key in:

- ✓ A formula or formulas
- ✓ A series of equations—and solve for all variables
- ✓ Constraints (like  $X$  has to be  $<$  or  $=$  2)
- ✓ A function to plot
- ✓ Unit conversions
- ✓ Maximization and minimization problems
- ✓ Interest Rate/Present Value calculations
- ✓ Variables we call "What happens?," like "What happens if I change this variable to 21 and that variable to 27?"

**All this power for only \$99.95!**

Equation-solving used to be a mainframe problem, but we've solved that problem.

Eureka: The Solver is all you need—and it's yours for only \$99.95!

That kind of savings you can calculate with your fingers!

## System requirements

IBM PC, AT, XT, Portable, 3270 or true compatibles.  
PC-DOS (MS-DOS) 2.0 and later. 384K.

\*Introductory price—good through July 1, 1987

**Only \$99.95!\***



**BORLAND**  
INTERNATIONAL

*Vive la différence*

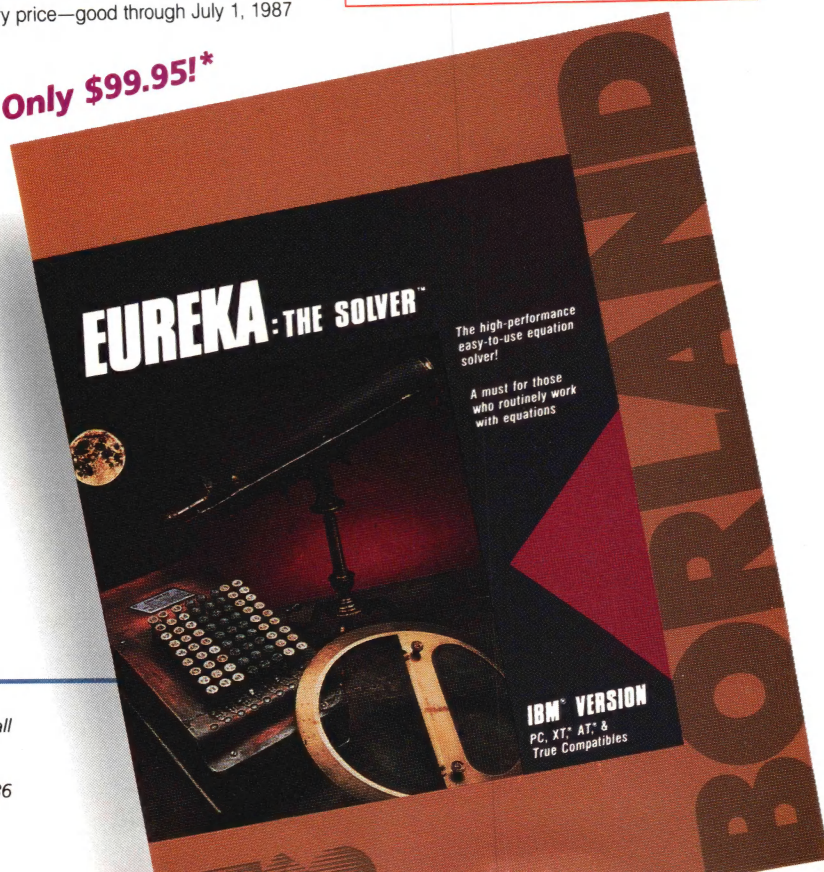
4585 SCOTTS VALLEY DRIVE  
SCOTTS VALLEY, CA 95066  
(408) 438-8400 TELEX: 172373

For the dealer nearest you or to order by phone call

**(800)255-8008**

in CA (800) 742-1133 in Canada (800) 237-1136

BI-1103A



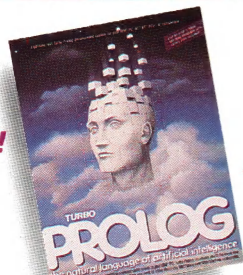


# Turbo Prolog™

“ If you're at all interested in artificial intelligence, databases, expert systems, or new ways of thinking about programming, by all means plunk down your \$100 and buy a copy of Turbo Prolog.

**Bruce Webster, BYTE** ”

**Only  
\$99.95!**



Turbo Prolog, the natural language of Artificial Intelligence, is the most popular AI package in the world with more than 100,000 users. It's the 5th-generation computer programming language that brings supercomputer power to your IBM PC and compatibles. You can join the AI revolution with Turbo Prolog for only \$99.95. Step-by-step tutorials, demo programs and source code included.

## New! Turbo Prolog Toolbox

Our new Turbo Prolog Toolbox™ enhances Turbo Prolog—with more than 80 tools and over 8,000 lines of source code that can easily be incorporated into your programs. It includes about 40 example programs that show you how to use and incorporate your new tools.

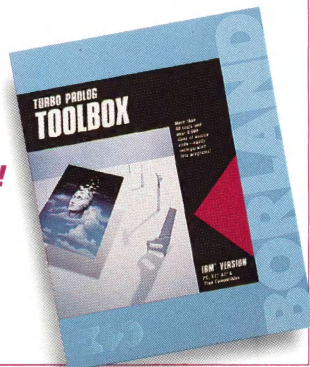
New Turbo Prolog Toolbox features include:

- ✓ Business graphic generation
- ✓ Complete communications package
- ✓ File transfers from Reflex, dBASE III, 1-2-3, Symphony
- ✓ A unique parser generator
- ✓ Sophisticated user-interface design tools

It's the complete developer's toolbox and a major addition to Turbo Prolog. You get a wide variety of menus—pull-down, pop-up, line, tree and box—so you can choose the one that suits your application best. You'll quickly and easily learn how to produce graphics; set up communications with remote devices; read information from Reflex,\* dBASE III,\* Lotus 1-2-3\* and Symphony\* files; generate parsers and design user interfaces. All of this for only \$99.95.

**NEW**

**Only  
\$99.95!**



### System requirements

Turbo Prolog: IBM PC, XT, AT or true compatibles. PC-DOS (MS-DOS) 2.0 or later. 384K. Turbo Prolog Toolbox requires Turbo Prolog 1.10 or higher. Dual-floppy disk drive or hard disk. 512K.

MACINTOSH™  
VERSION ALSO  
AVAILABLE

# Turbo Pascal®

The power and high performance of Turbo Pascal is already in the hands of more than half-a-million people. The technically superior Turbo Pascal is the *de facto* worldwide standard and the clear leader.

The Turbo Pascal family includes:

- Turbo Pascal® 3.0
- Turbo Tutor® 2.0
- Turbo Database Toolbox®
- Turbo Editor Toolbox®
- Turbo Graphix Toolbox®
- Turbo GameWorks®
- Turbo Pascal Numerical Methods Toolbox™



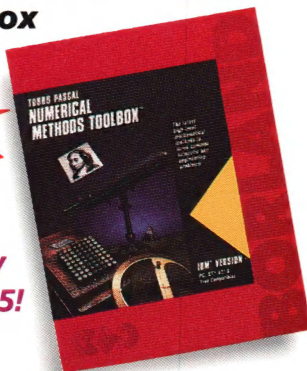
Turbo Pascal, the worldwide standard in high-speed compilers, and family.

“ The language deal of the century.  
**Jeff Duntemann, PC Magazine** ”

## New! Turbo Pascal Numerical Methods Toolbox

**NEW**

**Only  
\$99.95!**



What our new Numerical Methods Toolbox will do for you now:

- ✓ Find solutions to equations
- ✓ Interpolations
- ✓ Calculus: numerical derivatives and integrals
- ✓ Differential equations
- ✓ Matrix operations: inversions, determinants and eigenvalues
- ✓ Least squares approximations
- ✓ Fourier transforms

As well as a free demo FFT program, you also get Least Squares Fit in 5 different forms:

1. Power
2. Exponential
3. Logarithm
4. 5-term Fourier
5. 5-term Polynomial

They're all ready to compile and run.

**All this for only \$99.95 !**

### System requirements

IBM PC, XT, AT or true compatibles. PC-DOS (MS-DOS) 2.0 or later. Turbo Pascal 2.0 or later. Graphics module requires graphics monitor with IBM CGA, IBM EGA, or Hercules compatible adapter card, and requires Turbo Graphix Toolbox. 8087 or 80287 numeric co-processor not required, but recommended for optimal performance. 256K.

### Turbo Pascal 3.0.

Includes 8087 & BCD features for 16-bit MS-DOS and CP/M-86 systems. CP/M-80 version minimum memory: 48K; 8087 and BCD features not available. 128K.





# Turbo Basic<sup>®</sup>

NOW SHIPPING!

## Introducing Turbo Basic, the high-speed BASIC you'd expect from Borland!

It's the BASIC compiler you've been waiting for. And it's so fast that you'll never have to wait again.

Turbo Basic is a complete development environment; it includes a lightning-fast compiler, an interactive editor, and a trace debugging system.

Because Turbo Basic is compatible with BASICA, chances are that you already know how to use Turbo Basic.

## With Turbo Basic your only speed is "Full Speed Ahead"!

You probably already know us for both Turbo Pascal<sup>®</sup> and Turbo Prolog.<sup>™</sup> Well, we've done it again!

We created Turbo Basic, because BASIC doesn't have to be slow.

In fact, building fast compilers is a Borland specialty; both our Turbo Pascal and our Turbo Prolog outperform all their rivals by factors, and with Turbo Basic, we're proud to introduce the first high-speed BASIC compiler for the IBM<sup>®</sup>PC. If BASIC taught you how to walk, Turbo Basic will teach you how to run!

### The Critics' Choice

“Borland has succeeded in stretching the language without weighing us down with unnecessary details . . . Turbo Basic is the answer to my wish for a simple yet blindingly fast recreational utility language . . . The one language you can't forget how to use, Turbo Basic is a computer language for the masses, the masters, the masses, and me.

Steve Gibson, InfoWorld

Borland's Turbo Basic has advantages over the Microsoft product, including support of the high-speed 8087 math chip.

John C. Dvorak

## Turbo Basic ends the basic confusion

There's now one standard: Turbo Basic.

It's fast, BASICA-compatible, and because Turbo Basic is a Borland product, the price is right, the quality is there, and the power is at your fingertips. You see, Turbo Basic's part of the fast-growing Borland family of programming languages—we call it the "Turbo Family." Hundreds of thousands of users are already using Borland's languages, so you can't go wrong. So join a whole new generation of smart IBM PC users—get your copy of Turbo Basic today. You get an easy-to-read 300+ page manual, two disks, and a free MicroCalc spreadsheet—and an instant start in the fast new world of Turbo Basic. All of this for only \$99.95—Order your copy of Turbo Basic today!

### Free spreadsheet included, complete with source code!

Yes, we've included MicroCalc, our sample spreadsheet, complete with source code, so that you can get started right away with a "real program." You can compile and run it "as is," or modify it.

### A technical look at Turbo Basic

- ✓ Full recursion supported
- ✓ Standard IEEE floating-point format
- ✓ Floating-point support, with full 8087 (math co-processor) integration. Software emulation if no 8087 present
- ✓ Program size limited only by available memory (no 64K limitation)
- ✓ EGA and CGA support
- ✓ Access to local, static, and global variables
- ✓ Full integration of the compiler, editor, and executable program, with separate windows for editing, messages, tracing, and execution
- ✓ Compile, run-time, and I/O errors place you in the source code where error occurred
- ✓ New long integer (32-bit) data type
- ✓ Full 80-bit precision
- ✓ Pull-down menus
- ✓ Full window management

### System requirements

IBM PC, XT, AT and true compatibles, PC-DOS (MS-DOS) 2.0 or later. One floppy drive, 256K.

Only \$99.95!







# Turbo C<sup>®</sup>

## **Turbo C: The fastest, most efficient and easy-to-use C compiler at any price**

Compilation speed is more than 7000 lines a minute, which makes anything less than Turbo C an exercise in slow motion. Expect what only Borland delivers: Quality, Speed, Power and Price.

## **Turbo C: The C compiler for amateurs and professionals**

If you're just beginning and you've "kinda wanted to learn C," now's your chance to do it the easy way. Like Turbo Pascal, Turbo C's got everything to get you going.

If you're already programming in C, switching to Turbo C will considerably increase your productivity and help make your programs both smaller and faster. Actually, writing in Turbo C is a highly productive and effective method—and we speak from experience. Eureka: The Solver and our new generation of software have been developed using Turbo C.

## **Turbo C: a complete interactive development environment**

**Free MicroCalc spreadsheet with source code**

Like Turbo Pascal and Turbo Prolog, Turbo C comes

with an interactive editor that will show you syntax errors right in your source code. Developing, debugging, and running a Turbo C program is a snap.

## **Turbo C: The C compiler everybody's been waiting for. Everybody but the competition**

Borland's "Quality, Speed, Power and Price" commitment isn't idle corporate chatter. The \$99.95 price tag on Turbo C isn't a "typo," it's real. So if you'd like to learn C in a hurry, pick up the phone. If you're already using C, switch to Turbo C and see the difference for yourself.

### **System requirements**

IBM PC, XT, AT and true compatibles. PC-DOS (MS-DOS) 2.0 or later. One floppy drive. 320K.

## **Technical Specifications**

- ✓ **Compiler:** One-pass compiler generating linkable object modules and inline assembler. Included is Borland's high performance "Turbo Linker." The object module is compatible with the PC-DOS linker. Supports tiny, small, compact, medium, large, and huge memory model libraries. Can mix models with near and far pointers. Includes floating point emulator (utilizes 8087/80287 if installed).
- ✓ **Interactive Editor:** The system includes a powerful, interactive full-screen text editor. If the compiler detects an error, the editor automatically positions the cursor appropriately in the source code.
- ✓ **Development Environment:** A powerful "Make" is included so that managing Turbo C program development is highly efficient. Also includes pull-down menus and windows.
- ✓ **Links with relocatable object modules** created using Borland's Turbo Prolog into a single program.
- ✓ **ANSI C compatible.**
- ✓ **Start-up routine source code included.**
- ✓ **Both command line and integrated environment versions included.**

\*Introductory price—good through July 1, 1987

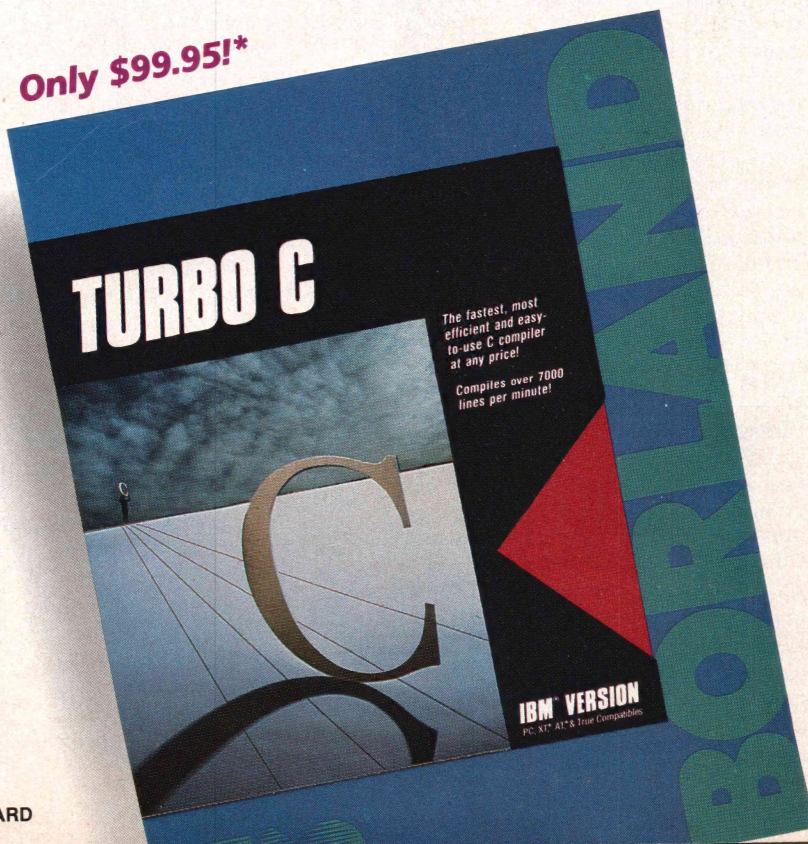
### **Sieve benchmark (25 iterations)**

	<b>Turbo C</b>	Microsoft® C	Lattice C
Compile time	<b>3.89</b>	16.37	13.90
Compile and link time	<b>9.94</b>	29.06	27.79
Execution time	<b>5.77</b>	9.51	13.79
Object code size	<b>274</b>	297	301
Price	<b>\$99.95</b>	\$450.00	\$500.00

Benchmark run on a 6 Mhz IBM AT using Turbo C version 1.0 and the Turbo Linker version 1.0; Microsoft C version 4.0 and the MS overlay linker version 3.51; Lattice C version 3.1 and the MS object linker version 3.05.

All Borland products are trademarks or registered trademarks of Borland International, Inc. or Borland Analytica, Inc. Other brand and product names are trademarks or registered trademarks of their respective holders. Copyright 1987 Borland International BI-11034

**Only \$99.95!\***



For the dealer nearest you, or to order by phone  
**call (800) 255-8008**  
CA (800) 742-1133  
Canada (800) 237-1136





## NEW! FROM BLAISE COMPUTING

Today's programmers need more than yesterday's tools. Requirements such as removable windows and "sidekickable" pop-up utilities are changing the face of program design. You need to filter interrupts so that other resident programs still work. You need the ability to switch between multiple display pages and monitors. Today's technical demands are almost endless, but C TOOLS PLUS gives you what you need.



## SOLID LIBRARY SUPPORT

Blaise Computing offers you solid library support that can meet all your demands and more. C TOOLS PLUS embodies the full spectrum of general-purpose utility functions that are critical to today's applications.

Here's just part of the PLUS in C TOOLS PLUS:

- ◆ **C TOOLS and C TOOLS 2** compatibility — two packages that receive rave reviews for quality, organization, usability and documentation.
- ◆ **FULL SOURCE CODE**

# C Tools Plus<sup>TM</sup> For The Programmer Whose Alphabet Begins & Ends With "C"



- ◆ **WINDOWS** that are stackable, removable, that support word wrap and that can accept user input.
- ◆ **INTERRUPT SERVICE ROUTINE** support for truly flexible, robust and polite resident applications.
- ◆ **MULTIPLE** monitor and display support, including EGA 43-line mode.
- ◆ **FAST DIRECT VIDEO ACCESS** for efficiency that will not constrain good program design.
- ◆ **DOCUMENTATION, TECHNICAL SUPPORT** and attention to detail that have distinguished Blaise Computing products over the years.

*C TOOLS PLUS supports the Microsoft (and IBM) 3.00 and Lattice 3.00 C compilers and is just \$175.00.*

Also Available Are:  
**C VIEW MANAGER** — A kit for building data entry screens and menus. Begin by designing on-screen what the operator will see; call upon our library functions from your program to display the screens and retrieve the data. Just \$275, including all library source code.

**C ASYNCH MANAGER** — provides the crucial core of hardware interrupt support needed to build applications that communicate. It

also includes the "XMODEM" file-transfer protocol and support for Hayes-compatible modems. All source code is included for \$175. **C TOOLS & C TOOLS 2** — an indispensable combination still available at a low price of \$175, including all source code. See review in PC Tech Journal, 6/85.

## BLAISE COMPUTING INC.

2560 Ninth Street, Suite 316 Berkeley, CA 94710 (415) 540-5441

**ORDER TOLL-FREE 800-227-8087!**

YES, send me the PLUS I need! Enclosed is \$\_\_\_\_\_ for C TOOLS PLUS. (CA residents add 6½% Sales Tax. All domestic orders add \$10.00 for Federal Express shipping.)

Name: \_\_\_\_\_ Phone: (\_\_\_\_) \_\_\_\_\_

Shipping Address: \_\_\_\_\_ State: \_\_\_\_\_ Zip: \_\_\_\_\_

City: \_\_\_\_\_ Exp. Date: \_\_\_\_\_

VISA or MC #: \_\_\_\_\_

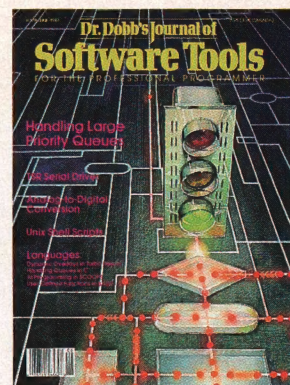


## ARTICLES

- Efficient queue control** ► **ALGORITHMS: An Efficient Algorithm for Large Priority Queues** 16  
by Robert Jay Brown  
By deferring decisions about lower-level priorities, this algorithm is able to assign the top priority more efficiently.
- Mathematical basis for a space-saving trick** ► **COMMUNICATIONS: Two-Bit Analog-to-Digital Conversion** 22  
by John Musselman  
How to use hardware interrupts and a demonstration of techniques common in real-time programming
- TSR serial driver** ► **ALGORITHMS: The XOR Chain** 28  
by David E. Cortesi  
The Swapped-Out Intern returns with a technique that exploits a curious property of doubly linked lists.
- Handling large Turbo Pascal programs** ► **COMMUNICATIONS: An Extended COM Port Driver** 42  
by Thomas A. Zimniewicz  
Excom is a terminate-and-stay-resident program that gives the IBM PC interrupt-driven buffered input with flow-control selection and support for higher baud rates.
- Unix telecommunications software** ► **LANGUAGES: Dynamic Memory Overlays for Turbo Pascal** 50  
by Steve McMahon  
Steve shows how to get around the 64K limit on executable code imposed by Turbo Pascal without resorting to slow disk overlays.
- Queue control in C** ► **COMMUNICATIONS: A Unix BBS Using Shell Scripts** 54  
by Jan L. Harrington  
Jan shows just how she wrote a bulletin board system using Unix V Bourne shell scripts and XMODEM.

## COLUMNS

- For the programmer's bookshelf** ► **C CHEST** 102  
by Allen Holub  
Allen delves into the subject of priority queues and, in Flotsam and Jetsam, discusses standard *#include* files.
- Object-oriented programming** ► **16-BIT SOFTWARE TOOLBOX** 112  
by Ray Duncan  
Ray discusses programming books. He also offers a poor man's MAKE utility and an MS-DOS programming tip.
- Time to hock the Volvo** ► **ARTIFICIAL INTELLIGENCE** 116  
by Ernest R. Tello  
Ernie presents some examples of object-oriented programming using SCOOPS, an extension of PC Scheme, "the Turbo Pascal of the PC LISP family."
- What's right with high-level languages?** ►
- | FORUM                  |            | PROGRAMMER'S SERVICES                                 |
|------------------------|------------|---|
| <b>EDITORIAL</b>       | <b>6</b>   | <b>THE STATE OF BASIC:</b> 128                        |
| by Michael Swaine      |            | A comparison of user-defined functions in four BASICS |
| <b>RUNNING LIGHT</b>   | <b>8</b>   | <b>OF INTEREST:</b> 132                               |
| by Levi Thomas         |            | Products for programmers                              |
| <b>ARCHIVES</b>        | <b>8</b>   | <b>ADVERTISER INDEX:</b> 113                          |
| <b>LETTERS</b>         | <b>10</b>  | Where to find those ads                               |
| by you                 |            |   |
| <b>VIEWPOINT</b>       | <b>14</b>  |   |
| by Brian R. Anderson   |            |   |
| <b>SWAINE'S FLAMES</b> | <b>136</b> |   |
| by Michael Swaine      |            |   |



## About the Cover

Handling the traffic flow of multi-tasking requires an efficient queueing algorithm such as the one that has just given the green light to a task in this month's cover illustration.

## This Issue

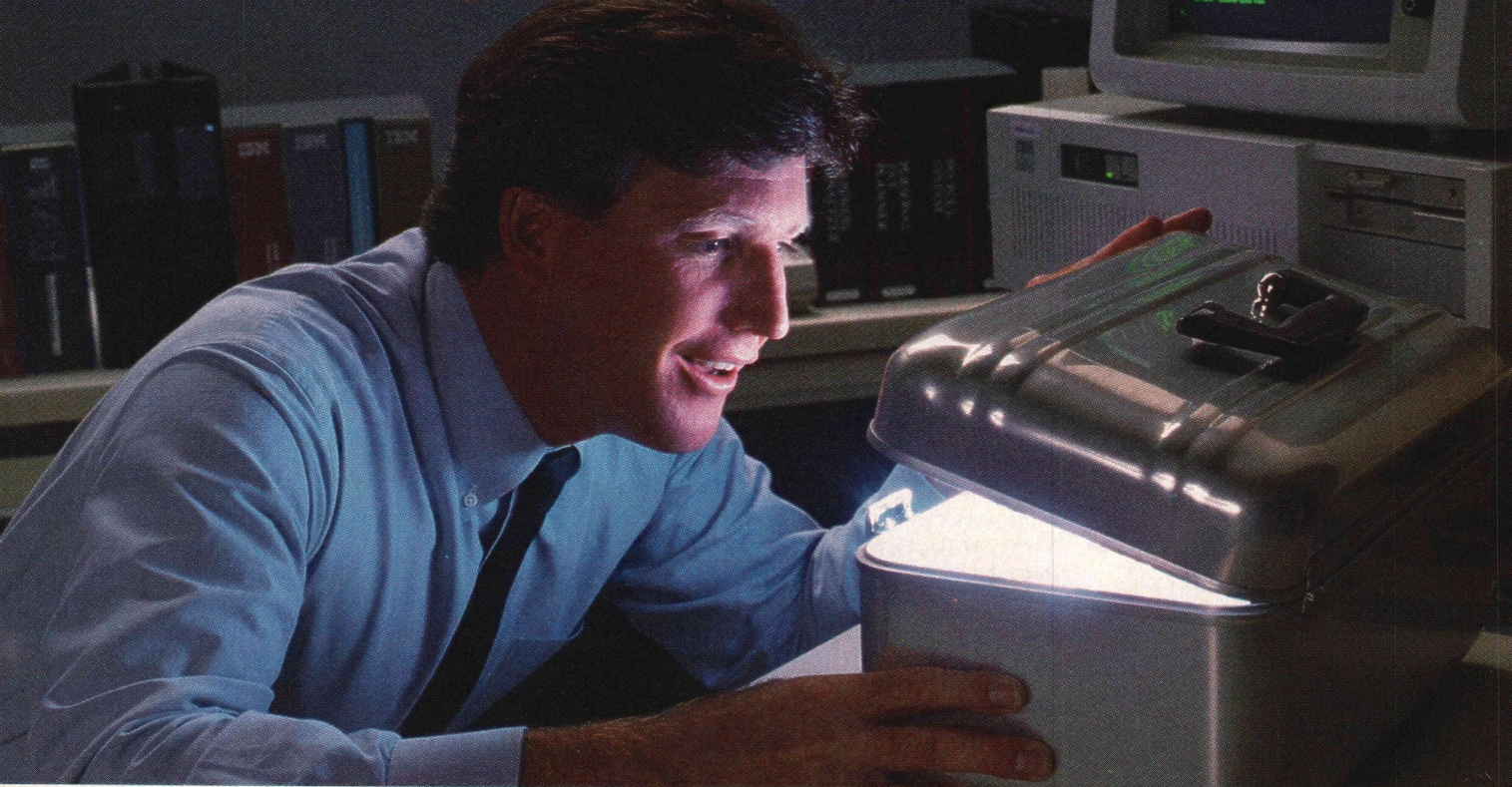
If this issue looks packed, it is. Three articles deal with communications techniques. Robert Jay Brown and Allen Holub discuss algorithms for priority queues. Dave Cortesi, a longtime *DDJ* columnist, contributes a rare bit on doubly linked lists. Steve McMahon gives aid and comfort to serious Turbo Pascal programmers; Ernie Tello and Allen Holub give tips to new PC Scheme and C programmers, respectively; and Forth, assembly-language, Modula-2, and BASIC programmers will all find something of interest in this issue.

## Next Issue

So you really did it? You sold the Volvo and bought a 386 machine and now you're going to develop software for it? Only you don't want to wait for Microsoft to deliver OS/2 sometime next year? We hope you saved \$2.95 for our July issue, in which we will discuss development tools for the 386 that are available today.



# Unleash The Most Powerful Development Tools On The Planet DOS.



## UNIFY DBMS/DOS. The UNIX World Leader Brings A New Dimension To DOS Application Development.

What happens as the DOS world expands? As a new generation of hardware takes over? As networking becomes more important? The potential is enormous. But until now, the tools to achieve it have been limited.

Now a leader from another world unleashes that potential: UNIFY® DBMS. The leading relational DBMS in the UNIX™ world. And now, the most advanced set of application development tools in the DOS world.

With UNIFY DBMS, DOS developers have new power to build more sophisticated applications than ever before possible.

The power to write high performance "C" programs that will access the data base, using Unify's Direct Host Language Interface.

The power of an industry standard query language—SQL.

The power of unmatched speed in production applications. Only UNIFY DBMS is specifically engineered for transaction throughput. With unique performance features like PathFinder™ Architecture multiple access methods, for the fastest possible data base access.

The power of comprehensive program development and screen management tools. Plus a state-of-the-art fourth generation report-writer.

What's more, with UNIFY DBMS, the potential of networked applications becomes a reality. Unlike DBMS systems which were originally single-user (and which have a long stretch to accommodate more users), UNIFY DBMS is a proven multi-user system.

And because UNIFY DBMS/DOS is the best of two worlds, it offers you the most powerful benefit of all: DBMS applications that can grow as your needs grow. From single user DOS. To networked DOS. To multi-user UNIX. All without changing your applications.

**Call the Unify Information Hotline  
for our free booklet: The New DOS World.  
(503) 635-7777**



**UNIFY**  
CORPORATION

4000 Kruse Way Place  
Lake Oswego, OR 97034



# WHY LOGITECH MODULA-2 IS MORE POWERFUL THAN PASCAL OR C.

"A clear winner... The integrated editor is  
a joy to use." *BYTE Magazine,*  
Jan. '87

## APPRENTICE PACKAGE \$99

- Separate Compilation  
w/inter-module typechecking
- Native Code Generation
- Large Memory Model Support
- Most Powerful Runtime Debugger
- Comprehensive Module Library
- Maintainability
- Translator from Turbo and  
ANSI Pascal

## WIN A FREE TRIP TO Switzerland



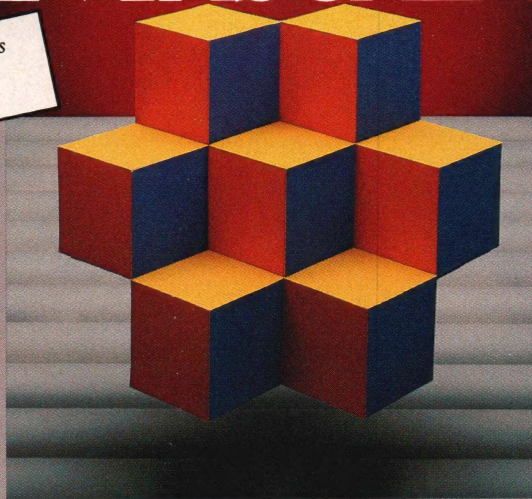
### HOMELAND OF MODULA-2

Return your Modula-2 Registration Card or a reasonable facsimile\* postmarked between March 1, 1987 and May 31, 1987 to be included in a once-only drawing!

**Grand Prize:** One week excursion for 2 in Zurich, Switzerland including a guided tour of ETH, the University where Modula-2 was created by Niklaus Wirth. European customers may substitute a trip to Silicon Valley, California.

**Second and Third Prizes:** LOGITECH C7 Mouse or LOGITECH Bus Mouse with Paint & Draw software—a \$219 value, absolutely free!

\*Write to Logitech, Inc. for a registration card facsimile.



## WIZARDS' PACKAGE \$199

**NEW!**

### APPRENTICE PACKAGE \$99

Everything you need to begin producing reliable maintainable Modula-2 code. Includes the Compiler with 8087 support, integrated Editor, Linker, and BCD Module. We're also including FREE our Turbo Pascal to Modula-2 Translator!

**NEW!**

### WIZARDS' PACKAGE \$199

This package contains our Plus Compiler—for professional programmers or for those who just want the best. The Plus Compiler with Integrated Editor requires 512K and takes advantage of the larger memory to increase compilation speed by 50%. Our Turbo Pascal to Modula-2 Translator is also included at no extra charge.

**NEW!**

### MAGIC TOOLKIT \$99

We've put our most powerful development tools into one amazing Toolkit for use with either the Apprentice or Wizards' packages. Highlighted by our Runtime Debugger, the finest debugging tool available anywhere, the Toolkit also includes our Post Mortem Debugger, Disassembler, Cross Reference utility and Version which keeps track of different versions of one program. Our MAKE Utility figures out module dependencies and automatically selects those affected by code changes to minimize recompilation and relinking. We also provide source code of our major library modules for you to customize—or just play with.

### WINDOW PACKAGE \$49

Now you can build true windowing into your Modula-2 code. Features virtual screens, color support, overlapping windows and a variety of borders.

### ROM PACKAGE AND CROSS RUN TIME DEBUGGER \$299

For those who want to produce rommable code. You can even debug code running in ROM from your PC.

Turbo Pascal is a registered trademark of Borland International.

Call for information about our  
VAX/VMS version, Site License, University  
Discounts, Dealer & Distributor pricing.

To place an order call  
toll-free:

**800-231-7717**

In California:

**800-552-8885**

## YES! I want the spellbinding power of LOGITECH Modula-2!

- |   |              |
|---|--------------|
| <input type="checkbox"/> Apprentice Package | <b>\$99</b>  |
| <input type="checkbox"/> Wizards' Package   | <b>\$199</b> |
| <input type="checkbox"/> Magic Toolkit      | <b>\$99</b>  |
| <input type="checkbox"/> Window Package     | <b>\$49</b>  |
| <input type="checkbox"/> ROM Pkg/Cross RTD  | <b>\$299</b> |

Add \$6.50 for shipping and handling. Calif. residents  
add applicable sales tax. Prices valid in U.S. only.

Total Enclosed \$ \_\_\_\_\_

☐ VISA ☐ MasterCard ☐ Check Enclosed

Card Number \_\_\_\_\_ Expiration Date \_\_\_\_\_

Signature \_\_\_\_\_

Name \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_

Zip \_\_\_\_\_ Phone \_\_\_\_\_



## LOGITECH

LOGITECH, Inc.  
805 Veterans Blvd. Redwood City, CA 94063  
Tel: 415-365-9852

**In Europe:**

LOGITECH SA, Switzerland  
Tel: 41-21-879656 • Telex 458 217 Tech Ch

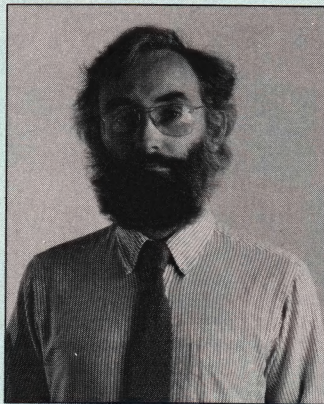
**In Italy:**

Tel: 39-2-215-5622



## EDITORIAL

The summer of 1987. Midyear. Time to take stock; time to assess what we've seen of 1987 and mull over what it all means. What can we say? That it was the spring of hype, the summer of virtuality? I think it's a watershed summer, a good time to be a software developer.



the Fifth Generation Project, and it's also going to have a significant impact on software developers by providing an entirely new, coherent, Japanese-script-based programming platform.

These new environments and new tools speak of new opportunities for the software

developer. But they speak in the ambient din, and it's easy to miss their message.

If you're currently evaluating the software development opportunities presented by these recent advances, could I make some modest suggestions?

Don't listen too closely to the marketers. Marketing experts are magicians; when it's difficult to create value, marketers can create need. They can convince people that there is something terribly wrong with them that only your product can cure. But it's so easy to create real value with a piece of software that I'm just not sure that we need the digital deodorant. If the product is good, marketing it becomes a simpler matter.

And I wouldn't listen too closely to users, heretical as that statement may be. Listening to users doesn't generally contribute a lot to creating something new. User feedback is useful when you're tweaking existing products. Users may not know anything about software, but they know what they don't like.

Ultimately, I argue, the real breakthroughs come from listening to the technology. That should be particularly true at a technological watershed like this summer.

Hock the Volvo, buy that 68020 or 80386 machine, and start playing around.

*Michael Swaine*

Michael Swaine  
editor-in-chief

Microsoft, IBM, and Apple all claimed that we were entering a new generation of personal computing. Both Apple and IBM announced significant new machines. Both Apple and IBM committed to fast, smart new buses and more open architectures, which will create new challenges and opportunities for hardware and software developers.

Microsoft revealed plans for getting parts of its OS/2 for 80286 and 80386 processors into developers' hands this year. Ever since Gary Kildall put an operating system on an 8080, the only arguably revolutionary advance in personal computer operating systems has been the Macintosh system software. It might be a stretch to suggest that OS/2 could be the next, but when it really arrives it should prove fertile ground for software development.

Both Unix and C took steps toward standardization in the first half of this year, and the crowded C compiler market ripened toward some sort of mitosis. Both Unix and C presented new tools and new environments for development. Version management and CASE became more real for personal computer-based development.

## Dr. Dobb's Journal of Software Tools

FOR THE PROFESSIONAL PROGRAMMER

### Editorial

<b>Editor-in-Chief</b>	Michael Swaine
<b>Editor</b>	Tyler Sperry
<b>Managing Editor</b>	Vince Leone
<b>Assistant Editors</b>	Sara Noah Ruddy Levi Thomas Allen Holub
<b>Technical Editor</b>	Nick Turner
<b>Consulting Editor</b>	Ray Duncan
<b>Contributing Editors</b>	Michael Ham Bela Lubkin Namir Shammass Ernest R. Tello
<b>Copy Editor</b>	Rhoda Simmons
<b>Production</b>	
<b>Production Manager</b>	Bob Wynne
<b>Art Director</b>	Michael Hollister
<b>Assoc. Art Director</b>	Joe Sikoryak
<b>Technical Illustrator</b>	Frank Pollifrone
<b>Cover Artist</b>	Barron Storey
<b>Circulation</b>	
<b>Circulation Director</b>	Maureen Kaminski
<b>Newsstand Sales Mgr.</b>	Stephanie Ericson
<b>Book Marketing Mgr.</b>	Jane Sharninghouse
<b>Circulation Coordinator</b>	Kathleen Shay
<b>Administration</b>	
<b>Finance Director</b>	Kate Wheat
<b>Business Manager</b>	Betty Trickett
<b>Accounts Payable Supv.</b>	Mayda Lopez-Quintana
<b>Accts. Receivable Supv.</b>	Laura Di Lazzaro
<b>Advertising Director</b>	
<b>Ferris Ferdon</b>	(415) 366-3600
<b>Account Managers</b>	
<b>Lisa Boudreau</b>	(415) 366-3600
<b>Martha Brandt</b>	(415) 366-3600
<b>Gary George</b>	(404) 897-1923
<b>Michael Wiener</b>	(415) 366-3600
<b>Cynthia Zuck</b>	(718) 499-9333
<b>Promotions/Srvcs. Mgr.</b>	Anna Kittleson
<b>Advertising Coordinator</b>	Charles Shively

### M&T Publishing Inc.

<b>Chairman of the Board</b>	Otmar Weber
<b>Director</b>	C. F. von Quadt
<b>President and Publisher</b>	Laird Foshay
<b>Associate Publisher</b>	Michael Swaine

**Dr. Dobb's Journal of Software Tools** (USPS 307690) is published monthly by M&T Publishing Inc., 501 Galveston Dr., Redwood City, CA 94063; (415) 366-3600. Second-class postage paid at Redwood City and at additional entry points.

**Article Submissions:** Send manuscripts and disk (with article and listings) to the Editor.

**DDJ on CompuServe:** Type GO DDJ

**Address Correction Requested:** Postmaster: Send Form 3579 to *Dr. Dobb's Journal*, P.O. Box 27809, San Diego, CA 92128. **ISSN 0888-3076**

**Customer Service:** For subscription problems call: outside CA (800) 321-3333; in CA (619) 485-9623 or 566-6947. For book/software order problems call (415) 366-3600.

**Subscriptions:** \$29.97 per 1 year; \$56.97 for 2 years. Canada and Mexico add \$27 per year airmail or \$10 per year surface. All other countries add \$27 per year airmail. Foreign subscriptions must be prepaid in U.S. funds drawn on a U.S. bank. For foreign subscriptions, TELEX: 752-351.

**Foreign Newsstand Distributor:** Worldwide Media Service Inc., 386 Park Ave. South, New York, NY 10016; (212) 686-1520 TELEX: 620430 (WUI).

Entire contents copyright © 1987 by M&T Publishing Inc. unless otherwise noted on specific articles. All rights reserved.



*Dr. Dobb's Journal of Software Tools* is published by M&T Publishing Inc. under license from People's Computer Company, 2682 Bishop Dr., Suite 107, San Ramon, CA 94583, a nonprofit corporation.



# E=M C AZTEC

Our thanks to NASA for supplying this computer enhanced ultraviolet photo taken by Skylab IV of a solar prominence reaching out 350,000 miles above the sun's surface

## Genius Begins With A Great Idea ...

### But The Idea Is Just The Beginning

What follows is the time consuming task of giving form and function to the idea.

That's why we concentrate on building into our software development systems functions and features that help you develop your software ideas in less time and with less effort.

We've started 1987 by releasing new versions of our MS-DOS, Macintosh, Amiga, ROM, and Apple II C development systems. Each system is packed with new features, impressive performance, and a little bit more genius.

### Aztec C86 4.1

New PC/MS-DOS • CP/M-86 • ROM

Superior performance, a powerful new array of features and utilities, and pricing that is unmatched make the new Aztec C86 the first choice of serious software developers.

#### Aztec C86-p Professional System . . . \$199

• optimized C with near, far, huge, small, and large memory + Inline assembler + Inline 8087/80287 + ANSI support + Fast Float (32 bit) + optimization options • Manx Aztec 8086/80x86 macro assembler • Aztec overlay linker (large/small model) • source level debugger • object librarian • 3.x file sharing & locking • comprehensive libraries of UNIX, DOS, Screen, Graphics, and special run time routines.

#### Aztec C86-d Developer System . . . \$299

• includes all of Aztec C86-p • Unix utilities make, diff, grep • vi editor • 6+ memory models • Profiler.

#### Aztec C86-c Commercial System. . . \$499

• includes all of Aztec C86-d • Source for library routines • ROM Support • CP/M-86 support • One year of updates.

### Aztec C86 Third Party Software

A large array of support software is available for Aztec C86. Call or write for information. The following is a list of the most requested products: Essential Graphics • C Essentials • C Utility Library • Greenleaf Com. • Greenleaf General • Halo • Panel • PC-lint • PforCe • Pre-C • Windows for C • Windows for Data C terp • db Vista • Phact • Plink86Plus • C-tree.

### CP/M • TRS-80 • 8080/Z80 ROM

C compiler, 8080/Z80 assembler, linker, librarian, UNIX libraries, and specialized utilities.

#### Aztec C II-c (CP/M-80 & ROM). . . . . \$349

#### Aztec CII-d (CP/M-80) . . . . . \$199

#### Aztec C80 (TRS-80 3&4) . . . . . \$199

### Aztec C68k/Am 3.4

New Amiga Release

Amiga user groups across the USA voted Aztec C68k/Am release 3.3 the best Software Development System for the Amiga. Release 3.4 is more impressive.

#### Aztec C68k/Am-p Professional . . . . . \$199

A price/feature/performance miracle. System includes: optimized C • 68000/680x0 assembler • 68881 support • overlay linker • UNIX and Amiga libraries • examples.

#### Aztec C68k/Am-d Developer . . . . . \$299

The best of Manx, Amiga, and UNIX. System includes: all of Aztec C68k/Am-p • the Unix utilities make, diff, grep and vi.

#### Aztec C68k/Am-c Commercial . . . . . \$499

Aztec C68k/Am-d plus source for the libraries and one year of updates.

### Aztec C68k/Mac 3.4

New Macintosh Release

For code quality, reliability, and solid professional features, Aztec C for the Macintosh is unbeatable. This new release includes features and functions not found in any other Macintosh C development system.

#### Aztec C68k/Mac-p Professional . . . . . \$199

• optimized C • 68000/680x0 assembler • 68881 support • overlay linker • UNIX and Macintosh libraries • examples.

#### Aztec C68k/Mac-d Developer . . . . . \$299

The best of Manx, Macintosh, and UNIX. System includes: all of Aztec C68k/Am-p • the Unix utilities make, diff, grep • vi editor.

#### Aztec C68k/Mac-c Commercial . . . . . \$499

Aztec C68k/Am-d plus source for the libraries and one year of updates.

### Aztec C65

New ProDOS Release

Aztec C65 is the only commercial quality C compiler for the Apple II. Aztec C65 includes C compiler, 6502/65C02 assembler, linker, library utility, UNIX libraries, special purpose libraries, shell development environment, and more. An impressive system.

#### Aztec C65-c Commercial . . . . . \$299

• runs under ProDOS • code for ProDOS or DOS 3.3

#### Aztec C65-d Developer . . . . . \$199

• runs under DOS 3.3 • code for DOS 3.3

### Aztec ROM Systems

6502/65C02 • 8080/Z80 • 8086/80x86 • 680x0

An IBM or Macintosh is not only a less expensive way to develop ROM code, it's better. Targets include the 6502/65C02, 8080/Z80, 8086/80x86, and 680x0.

Aztec C has an excellent reputation for producing compact high performance code. Our systems for under \$1,000 outperform systems priced at over \$10,000.

#### Initial Host Plus Target . . . . . \$750

#### Additional Targets . . . . . \$500

#### ROM Support Package . . . . . \$500

### Vax, Sun, PDP-11 ROM HOSTS

Call for information on Vax, PDP-11, Sun and other host environments.

### C' Prime

PC/MS-DOS • Macintosh

Apple II • TRS-80 • CP/M

These C development systems are unbeatable for the price. They are earlier versions of Aztec C that originally sold for as much as \$500. Each system includes C compiler, assembler, linker, librarian, UNIX routines, and more. Special discounts are available for use as course material.

#### C' Prime . . . . . \$75

### Aztec Cross Development Systems

Most Aztec C systems are available as cross development systems. Hosts include: PC/MS-DOS, Macintosh, CP/M, Vax, PDP-11, Sun, and others. Call for information and pricing.

### How To Become An Aztec C User

To become a user call 800-221-0440. From NJ or international locations call 201-542-2121. Telex: 4995812 or FAX: 201-542-8386. C.O.D., VISA, MasterCard, American Express, wire (domestic and international), and terms are available. One and two day delivery available for all domestic and most international destinations.

Aztec C is available directly from Manx and from technically oriented computer and software stores. Aztec Systems bought directly from Manx have a 30 day satisfaction guarantee.

Most systems are upgradable by paying the difference in price plus \$10. Site licenses, OEM, educational, and multiple copy discounts are available.

To order or for more information call today.

# 1-800-221-0440

In NJ or international call (201) 542-2121 • TELEX: 4995812

# MANX



# RUNNING LIGHT

**G**reetings, programmers. I'm Levi Thomas, assistant editor and host of DDJFORUM, our SIG on CompuServe. If my writing style seems conversational, it's because I'm accustomed to the SIG, on which my readers can talk back.

It's an interesting phenomenon, the on-line life. When I first began working for DDJ as electronic editor I felt like a DJ on the graveyard shift. Posting messages on the message board of the SIG, I got the eerie feeling that there was really nobody out there, that just maybe I was talking to myself. Luckily the FORUM members are a verbal bunch, and even the lurkers can be coaxed into responding if I ask the right questions or start a topic that gets their dander up.

Now I want to talk to you hard-copy readers. I wanna pull on yer coats about something. I hope you'll talk back to me, either on line or via the U.S. Snail.

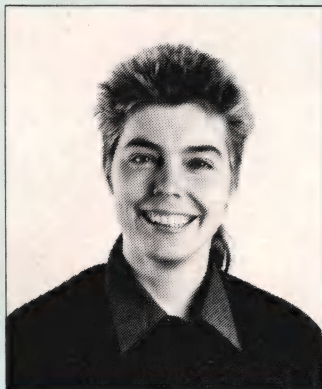
*<hopping onto soapbox>*

Microcomputers have grown up.

We all know the story of the microcomputer's Wonder Bread Years. Born with features only a mother could love (front panel toggle switches, 2K of RAM, no permanent storage, and so on), they have matured into smart, fast machines.

The microcomputer community has also grown up. What started a few years ago as a hobby has become a business. The community has become an industry. For many of you, your fascination with computers has turned into a shot at The Dream Job: A chance to make a living at something you love to do. A chance to make work into play.

However, like many new adults, this industry seems determined to disown all of its childlike qualities as



it seeks a grown-up identity. It's been giving up many of the values that fueled its growth. The Suits have moved in, and things are tightening up.

Examples: It's becoming next to impossible to get a job on pure ability; college degrees are the new prerequisite. The sense of community we once got from the computer magazines is disappearing. I see the term *hacker* changing—not evolving into the essence of what it originally meant but devolving into a derogatory term, a term feared by this new, "grown-up" industry. (Last year I was involved with the Hacker's Conference. Several computer businesses were nice enough to help fund the affair, but some of them asked that their names not be disclosed. They were nervous about having their company names associated with the word *hacker*.)

*<returning soapbox to closet>*

DDJ is still a reader-responsive magazine. If you would like to write for our October (Forth), November (graphics), or December (operating systems) issues, send us an outline. You can address your article proposals to our new editor, Tyler Sperry (you'll meet him on this page next month). If you just can't stand the thought of using the U.S. Snail, you can e-mail me on CompuServe (76703,4060), Usenet (well!levi), or arpanet (well!levi@lll-crg.arpa). We look forward to hearing from you.

Levi Thomas  
assistant editor

# ARCHIVES

## Inaugural Columns

"Welcome to the first edition of Programming Pastimes and Pleasures. Together we will explore the prosaic and fantastic possibilities of programming. Sometimes I will pose a problem to solve; sometimes I will discuss a programming technique or idea you may not have heard of; at other times I will tell you about a game you can play with your computer. At times I will be serious, and at other times I will be off-the-wall. Along the way, you will learn things you didn't know before, and you will find wonderful new ways to waste time with your computer. Most of all, I hope you will have fun."—Charles Wetherell, DDJ, January 1979.

"Doctor Dobb's Clinic is a new venture. It is a place for the display of techniques and discoveries. We want to cover any sort of method or trick you've found in your exploration of (for example, but not limited to): CP/M or OASIS or FLEX or APPLE DOS or TRSDOS or NEWDOS or RSTS or UNIX or UCSD PASCAL or....

"During our rounds at the clinic we may examine a compiler or an interpreter, revealing its bugs or showing how it can be made to perform better. We'd like to tell of unobvious uses for standard utilities. We want to uncover errors in published documentation, to warn people away from pitfalls, and to show off those 'eureka!' moments that make systems work rewarding."—Dave Cortesi (Resident Intern), DDJ, May 1981.

"Just after watching Doc Dobb's nationally televised speech from the NCC in Texas, I was surprised to receive a collect call from the Old Man himself. 'The lack of public domain software for the 16-bit microcomputers is appalling!' remarked our Fearless Leader. 'No one ants to shell out that kind of money just so they can stare at the operating system's sign-on message. I want you to institute a regular column in DDJ that will address the needs of 16-bit system users and promote the discussion and interchange of software.' Luckily, after laying down that sweeping and rather alarming mandate, the good Doctor had to hang up to go out on a house call (someone had choked on an Apple) and I was left to my own devices."—Ray Duncan, DDJ, September 1982.

DR. DOBB'S JOURNAL of  
**COMPUTER**  
Calisthenics & Orthodontia  
*Running Light Without Overbyte*



# Microsoft Avoids Challenge

We challenged Microsoft to a C compiler duel-to-the-finish, comparing compile, link and execution times, and we offered to stop advertising for two months if they won...

by Roy Sherrill, President, Datalight

Microsoft purchased our C-compiler during February 1987 and we still haven't heard from them. OK, Microsoft, we are extending our challenge deadline from April 1, 1987 to May 15, 1987. After all, the Microsoft ad claims "the fastest C you've ever seen." Your reply, Microsoft!

## Walter says Optimum-C is better

Walter Bright, the developer of Optimum C, says that Optimum C would win 7 out of 10 benchmarks as compared to Microsoft C, V.4.0. Walter explained to me that Optimum C includes a unique global optimizer that helps create compact code while increasing execution speed up to 30%. By the way, Borland, Walter is still waiting for his copy of Turbo C® V.1.0. Borland's ad claims "the fastest, most efficient and easy-to-use C compiler at any price."

After reviewing Borland's benchmarks, Walter claims that Optimum C is faster. And, as for ease of use, all Datalight C compilers have been shipped with a free Learn C program for the last six months. Also, our new EZ Interactive Editor will show you each syntax error in your source code, then compile or "make" and run your program, all from within the editor. OK, so let the Microsoft challenge begin...

## We only ask the following...

The benchmark suite will consist of the set of programs that Microsoft supplied to *Computer Language* for their February 1987 C compiler review issue. Microsoft will make available the programs to Datalight at least two weeks prior to the benchmarking. The benchmarking will be between Microsoft C 4.0 and Optimum-C. It will occur at a mutually agreed upon time and place. Interested individuals will be allowed to attend. The benchmarks will be compiled and run on a standard IBM PC-AT.

There will be two separate tests for each program: compile and link speed, and execution speed. For each test, a representative from each company will set up the compiler so that it performs at its best.

The benchmarks will be adjusted so that they take sufficiently long to run, that the tolerance involved in timing them is insignificant. The winner is determined by the compiler with the faster execution times for the majority of the benchmarks. We'd like an answer from Microsoft no later than May 15, 1987.

## So what's a global optimizer?

A global optimizer looks at an entire function at once, analyzing and optimizing the whole function. A technique called data flow analysis is used by Optimum-C to gather information about each function. This enables your compute-bound programs to execute as much as 30% faster after global

optimization. But, there is one catch...because the global optimizer ruthlessly searches for ways to speed-up execution speed and minimize memory usage, it has relatively slow compile times. No need to worry, though, because you can merely turn the global optimizer off. In fact, you can select all, none, or part of the following optimizations: constant propagation, copy propagation, dead assignment elimination, dead variable elimination, dead code elimination, do register optimizations, global common subexpression elimination, loop invariant removal, loop induction variables, optimize for space, optimize for time, and very busy expressions.

## Choose from five memory models

Speed your programs by selecting the memory model that best suits your application.

Model	Code	Data
Compact	64k total code & data	
Small	64k	64k
Program	1M	64k
Data	64k	1M
Large	1M	1M

## Compiling, one step...

Now with the one step DLC program you can create .OBJ, .EXE and .COM files. Also, DLC can handle multiple files and run MASM on your assembly files.

## Try Optimum-C risk free

Try Optimum-C for 30 days and if you are not 100% satisfied return it for a full refund. Also available is a C tutorial which is a combination workbook and floppy disk to help lead you through the C language with tutorials, quizzes, and program exercises.

O.K. Microsoft, it's up to you. We've put two months of advertising on the line that says you can't beat Optimum-C to a real test. Your answer, please?

## PRICES

Developer's Kit w/ C Tutorial	\$ 99
Optimum-C w/ C Tutorial	\$139
(both with library source)	

Add \$7 for shipping in US/\$20 outside US  
COD (add \$2.50)

Not Copy Protected

**ORDER TOLL-FREE TODAY!**

**1-800-221-6630**

## ATTENTION OEMs!

Contact us regarding arrangements.

Microsoft and MS-DOS are registered trademarks of the Microsoft Corporation. Turbo C is a registered trademark of Borland International.

## Magazine Reviewers Shocked by DATALIGHT's Performance...

"Reviewing this compiler was quite a surprise for us. For such a low price, we were expecting a "lightweight" compiler. What we got was a package that is as good as or better than most of the "heavyweights." Datalight C implements a complete C language. It also compiles quickly, doesn't take up much disk space, and looks impressive in the benchmarks."

*DR. DOBBS, August 1986*

"This is a sharp compiler!... what is impressive is that Datalight not only stole the compile time show completely, but had the fastest Fibonacci executable time and had excellent object file sizes to boot!"

*COMPUTER LANGUAGE, February 1986*

## Optimum-C Version 3.0

**NEW!**

**EZ Interactive Development Environment**

**NEW!**

**Inline 8087/80287 Math Support**

- Full UNIX System 5 C language plus ANSI extensions
- Fast/tight code via powerful optimizations including common sub-expression elimination
- DLC one-step compile/link program
- Multiple memory model support
- UNIX compatible library with PC functions
- Compatible with DOS linker and assembler
- Third-party library support
- Automatic generation of .COM files
- Supports DOS pathnames, wild cards, and Input/Output redirection
- Compatible with Lattice C version 3.x
- Interrupt handling in C
- Debugger support
- ROMable code support/start-up source

## MS-DOS® Support Features

- Mouse support
- Sound support
- Fast screen I/O
- Interrupt handler

## MAKE Maintenance Utility

- Macro definition support
- MS-DOS internal commands
- Inference rule support
- TOUCH date manager

## Tools in Source Code

- cat—UNIX style "type"
- diff—Text file differences
- fgrep—fast text search
- pr—Page printer
- pwd—Print working directory
- wc—Word count

# Datalight

17505-68th Avenue NE, Suite 304  
Bothell, Washington 98011 USA  
(206) 367-1803



## LETTERS



### What's Right with High-Level Languages?

Dear DDJ,

Mike Suman's criticisms of Modula-2 in the February 1987 Viewpoint call for some comments.

First, although many Modula-2 implementations are now available, based on Wirth's *Programming in Modula-2* (either the second or third edition), both the language specifications and the preferred standard libraries are still being fine-tuned by such bodies as M2WG, the Modula-2 Working Group of the British Standards Institution. As I understand the situation, draft proposals from BSI now await reaction and further input via the ISO. *MODUS Quarterly* (published by the Modula-2 Users' Association) has been covering this debate and provides a "democratic" forum for suggestions, including Wirth's own reactions. Mike should throw in his pen'n'orth before the stonecutters start chiseling. Contact George Symons, MODUS, P.O. Box 51778, Palo Alto, CA 94303; (415) 322-0547.

Second, I must defend Brian Anderson's use of the identifiers *FIRST* and *LAST* in X68000. Although in the final, published version these turn out to be nonce constants, during the development stage (and later when the MC68030 is incorporated!) who knows what values they might assume or where else they might occur?

Finally, you must really sympathize with Niklaus. You

want to keep a language "simple," "portable," and "small-core," yet hardly is the ink dry before committees emerge wanting to add their pet piece of FORTRAN or Ada. Where do you draw the line? Certainly array and record constants would be useful (as in Turbo but not standard Pascal) and may find their way into the canon. Modula-2 does have the useful, non-Pascal *Module Body* construct (a piece of code invoked just once when the module is first executed).

Mike's preferred tabular layout for setting up Brian Anderson's *Table68K* will be possible under my own forthcoming language called Modula 1-2-3.

Stan Kelly-Bootle

25 Parkwood Ave.

Mill Valley, CA 94063

*Stan Kelly-Bootle is the author of The Modula-2 Primer. —eds.*

Dear DDJ,

In his February 1987 Viewpoint, Mike Suman raises a couple of objections that he suggests show something wrong with Modula-2 and other high-level languages. I don't

think his objections are cogent, and I don't think they have anything to do with high-level languages.

His first complaint concerns the use of constants to replace Arabic integers. This is not peculiar to high-level languages but is a feature of most assembly languages, too, so whatever is wrong, it can't be a problem of Modula-2 or of high-level languages in general.

His particular complaint concerns the use of *FIRST* and *LAST* to replace 1 and 118, respectively; in the program on which he is commenting, they appear in the declaration:

```
Table68K : ARRAY[FIRST .. LAST] OF
                TableRecord;
```

This does look odd, but the usual use of constants in this situation would omit the constant *FIRST* and replace the constant *LAST* by something more informative, such as *NumberOfOpCodes*, so the declaration would look like this:

```
Table68K : ARRAY[1 .. NumberOfOp-
                codes] OF TableRecord;
```

I can't see why he would object to that. He does say, "no one can maintain that it is really easier, or safer, to change values in an early definition than it is to change them in the only place in which they are used later, when it is obvious what the effects of the change are going to be." But six months and five revisions later, how will you know that the number of opcodes is used only once in the program, and how easy will it be to find that one place?

His larger complaint is that Modula-2 contains no compact and readable way to initialize a complex table. But then, what language does?

There are really two points mixed up here. One is that Modula-2 provides no way to initialize variables other than by assignment statements; there are no facilities for de-



*A scarlet letter?*



# Architecture of Wendin's Concurrent I/O System

**Wendin's Operating System Toolbox makes it possible to write a concurrent driver for any I/O device.**

Wendin's Operating System Toolbox kernel is composed of three major components: the scheduler, the memory manager, and the I/O subsystem. This month we will show how the I/O subsystem works, at the QIO (Queued Input/Output) device driver level.

The I/O system supports concurrent and waited I/O operations from multiple processes simultaneously. Its architecture was derived from the VAX/VMS QIO system, which was developed by Digital Equipment Corporation with these ideas in mind. The whole QIO mechanism takes advantage of many of the kernel's capabilities, including ASTs (Asynchronous System Traps), and event flags (which are used to signal I/O completion).

Before a program can perform I/O operations on a device, it must assign a channel to the device. A channel is simply a number, similar to an MS-DOS file handle. The operating system uses an assigned channel number as an index into an array of channel control blocks (CCBs) to get information about the channel and the corresponding device.

After assigning a channel with the SYS\_\_ASSIGN system service, the user program can make calls to SYS\_\_QIO or SYS\_\_QIOW (for concurrent or waited I/O) through the INT FF software interrupt. The kernel calls the routine EXEQIO, which validates the caller's parameters and I/O privileges. If the call parameters are valid and the channel number corresponds to an assigned CCB, EXEQIO will deliver a KERNEL mode AST to the routine ASTQIO in the same process context. This allows the driver to run on its own stack, but in the context of the calling process. ASTQIO receives all information about the I/O operation in an IRP (I/O Request Packet), created by EXEQIO from the user program's parameters.

```
void ast_qio (packet)
  struct irp *packet;
{
  struct ccb *c;
  struct ucb *u;
  void (*driver)(struct irp *);

  c = &io_chantbl [packet->chan];
  u = &io_unittbl [c->unit];
  driver = u->driver;
  (*driver)(packet);
} /* ast_qio */
```

*Listing 1. How ASTQIO dispatches the device driver given a channel number.*

ASTQIO uses the channel number to find the corresponding CCB, and obtain a pointer to a data structure describing the device assigned to, in the format of a unit control block (UCB). The unit number is used as an index into an array of UCBs, much as the channel number is used as an index into an array of CCBs. The UCB contains information about a specific device, including what general type of device is being referenced (for example, a terminal, mailbox, or disk), and a pointer to the device driver code. This double indirection (the channel obtains a CCB, which in turn references a UCB, which finally references the device driver) allows several channels to be assigned to the same device at the same time, and for any number of UCBs to share the same device driver code (for example, the disk driver defines a UCB for each disk drive, but there is only one device driver for all of them).

Once ASTQIO has obtained the UCB for the requested device, it calls the device driver directly. The device driver is responsible for determining what to do based on the I/O function given in the IRP. The device driver will typically call the system BIOS, or interface directly to the hardware, to perform an I/O operation.

Finally, the kernel must signal the user process that the I/O is done. The SYS\_\_QIO services allow two

methods for this: an event flag can be set, or an AST can be executed when the operation finishes. The kernel routine STOPIO examines the IRP associated with an operation, and determines which event flag to set, and whether to queue an AST to the user process. Finally, the IRP is removed from the system I/O queues, and returned to the memory pool. Control is then passed back up the chain and returns to the user program.

A device driver can be written to talk to any device, based on standard models provided by the Operating System Toolbox kernel. Examples include mailboxes (functionally equivalent to UNIX pipes), terminals, disk drives, optical disks, or even expanded or extended memory. Basically, the minimum I/O functions that the driver needs to support are the read and write block functions. For some devices (like disks) there will be a distinction between virtual, logical, and physical I/O. For others (like mailboxes), all of these functions will perform the same operation. You can decide how you want a device driver to work, and it's easy to integrate a driver into the toolbox kernel.

Next month, we'll continue with the internals of the toolbox. If you'd like to learn more about operating system design and architecture, pick up a copy of Operating System Toolbox from Wendin and follow along.

**Operating System**

**Toolbox: \$99**

**Wendin, Inc.**

**P.O. Box 3888**

**Spokane, WA 99220**

**(509) 624-8088**



fining initial values at the point of declaration as in some other languages (although the compiler I use provides this as an extension). Unless the compiler is very clever, this may mean some unnecessary work at execution time, but this does not seem like a major problem and certainly not one inherent in high-level languages.

The second point is that it is tedious to initialize a large table (in the example Suman discusses, it consists of 118 records of 4 fields each) with assignment statements. But there are other ways. For instance, I like to write a table in a readable form in an ASCII disk file and read it at run time to initialize the table. It's hard to beat this for readability, the table can be edited without recompiling the program, and the code to read the table is usually not difficult to write. There is a problem with reading in enumerated types from an ASCII file: because it can be a nuisance to translate the ASCII version of the identifiers for the values, you can compromise by using Arabic numbers for their ordinal values instead. This can be endured because the table file can easily contain a key for the values.

Suman offers an assembly-language version of the table in which sets are initialized by 16-digit binary numbers. I suppose it's a matter of taste, but I don't find 118 entries like that very readable, and I'm sure I'd make plenty of errors typing them into my program. And what would he do if he had a table with real numbers in it?

I suppose it would be nice if we could write our tables in readable tabular form in our code and have the compiler do the work, but there are lots of tasks we might like our compilers to perform, and we can't build them all in. I would guess that initializing complex tables is one of those tasks that is just as well left out of general-purpose languages.

John G. Bennett  
301 Roslyn St.  
Rochester, NY 14619-1813

Dear DDJ,

Though the Viewpoint by Mike Suman is interesting and raises some valid points, I find it to be misleading

in three specific ways.

First, the point about the constants *FIRST* and *LAST* being used only once is not quite right. Consider the index variable *i*. The initialization and manipulation of *i* should always be in the range *FIRST* . . . *LAST*, and further, it should exactly cover this range. The program is poorly written in that the index should have been initialized to *FIRST*, not 1 explicitly. Next, the value of the index at the end of the initialization should have been checked to see that it was equal to *LAST*, so that you know all the elements of the table have been initialized. Far from pointing out a place in which Modula-2 is too strict, this points out a place in which the programmer just didn't use the language properly.

Second, the fact that Modula-2 doesn't allow the specification of constant arrays is not an indication that Modula-2's type checking is too strict or that its model of programming is inadequate. It means that Modula-2 doesn't have this feature. It is clear that allowing constant aggregates to be uttered in the language is an independent issue from whether the language is inconveniently strict or not. Strictness and features provided are two completely different issues.

Third, asking if imagined (or even real) problems with the Modula-2 language mean there is "something wrong with the direction in which we are being led" is to miss the point altogether. The point of "higher-level" languages that have a strict type structure is not to prevent programmers from full expression or to make it harder to write legal programs. The point is to help programmers by making it easier for them to tell when they have written a meaningless or illegal program. Here we see that a programmer misused the constant definition facility of the language and that the language lacks the ability to utter constant-valued aggregates. But neither of these faults is a fault of strict type checking, nor of modularity, nor of the high-level nature of the language.

So, if we are being led in the direction of modularity and static type safety, Mike hasn't even begun to

present evidence that the direction is wrong. He has shown that small languages often lack features that are awkward to do without, and he has shown that people often forget that index variables ought to be initialized and manipulated using the same bounds that were used to declare the array they index into. These are indeed things to be wary of, and they show that Modula-2 is neither omnipotent nor errorproof. But this is no reason to throw away the baby of type checking and modularity with the bathwater of a possibly too small language and human error.

Wayne Throop  
86 Fearrington  
Pittsboro, NC 27312

## Happy Ducks

Dear DDJ,

Thank you for the very funny February cover revealing the true nature of us WordStar-imprinted programmers. But no thanks for the text editors article. For a DDJ feature article, it was amazingly uninformative—simply an excuse for a two-barrel discharge against us happy ducks.

Especially unfair was the statement that WordStar-style editors "usually use 'weird' file formats that can't be read by any other editor without some sort of conversion." WordStar itself works in straight ASCII when in its program-editing (non-document) mode. But that is indeed not the most usual WordStar-style editor. The MUWSSE is, of course, the Turbo Pascal editor, which many waterfowl use with other compilers as well. And that is nothing but straight ASCII. So, just what were you talking about?

Another gross misstatement was the wish list. Every programmer I know, even strictly dry land, will tell you that wish number 1 is speed, 2 is speed, and 3 is speed. Other wishes appear only after pausing for breath. No mention of that in your list. Of course, again, the MUWSSE—the Turbo editor—is the fastest thing under ten fingers. And for the price of just about any dedicated programming editor, you can get Turbo Pascal and SuperKey, and then you have the

(continued on page 122)



# Clarify and document your source listing and get an "organization chart" of your program's structure with two NEW utilities from Aldebaran Laboratories, for C, BASIC, Pascal, dBASE®, Fortran and Modula-2 programmers.

Now works with Fortran

"Occasionally, a utility comes along that makes a programmer's life much easier. SOURCE PRINT is such a program. It contributes to the programmer's job by organizing code into a legible format and by helping to organize the documentation and debugging process."

— PC Magazine  
Sept. 16, 1986

Source Print and Tree Diagrammer both have easy-to-use menus with point-and-shoot file selection, and let you search for files containing a given string. For IBM PC and compatibles with 256K.

Join thousands of programmers who are working more efficiently using Source Print and Tree Diagrammer. Order these indispensable tools today. We ship immediately, and there's no risk with our 60-day money-back guarantee. **Order both and save. Only \$125.00.**

**800-257-5773** Dept. 56  
In California:

**800-257-5774** Dept. 56

MasterCard, VISA, American Express, COD. Add \$5 for shipping/handling.

or see your local dealer!

Source Print and Tree Diagrammer are trademarks of Aldebaran Labs. dBASE is a trademark of Ashton Tate. Prices subject to change without notice.

## Source Print™

organizes your source code, simplifies debugging, and makes documentation a snap! It lists one or more source files with informative page headings and optional line numbers, while offering invaluable features:

**The Index** (Cross-Reference list) saves you time by showing exactly where variables are used and where functions, procedures, and routines are called.

**\$75<sup>00</sup>**

Locations where new values may be assigned to variables are shown, making it easy to track down that mysterious value change.

**Structure Outlining** solves the problem of hard-to-see nested control structures by automatically drawing lines around them.

**Automatic Indentation** of source code and listings reduces your editing time and ensures indentation accuracy.

**Plus . . .** Source Print generates a table of contents listing functions and procedures. Keywords can be printed in boldface on most printers. Multi-statement BASIC lines can be split for readability. Functions and procedures can be drawn by name from one or more source files to form a new file.

Before

```

1 source ()
2 while (lar < nres && ares(lar)[0] <= c)
3 {
4   if ((d = ares(lar)[1]) == 0)
5   {
6     p = &ares(lar)[1];
7     while (d <= ap)
8     {
9       p++;
10      loop++;
11    }
12    lar++;
13  }
14 }
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

Before

After

```

150 FOR INDEX = 1 TO 100
160 IF TB(INDEX) = 0 THEN X = 5
170 C = 50: WHILE K <= 1000: TB(K) = 0: K = K + X: WEND
180 GOSUB 2000
190 XT(C) = X: T2(C) = K: C = C + 1
200 NEXT INDEX

```

```

150 FOR INDEX = 1 TO 100
160 IF TB(INDEX) = 0 THEN X = 5
170 C = 50
180 WHILE K <= 1000
190   TB(K) = 0
200   K = K + X
210 GOSUB 2000
220 XT(C) = X
230 T2(C) = K
240 C = C + 1
250 NEXT INDEX

```

BASIC

Wed 12-31-86 07:22:03 INDEX (Cross Ref)  
all identifiers

inrecord	4.191	9.396	19.825	19.826
	21.889	22.922	22.953	23.978
	23.990			
ina	53.2293	53.2309	53.2319	53.2325
	54.2331	54.2332	54.2336	54.2346
	54.2354	54.2364	54.2365	54.2366
intext	4.193	9.395	43.1796	43.1815
	43.1820	45.1902		

Index

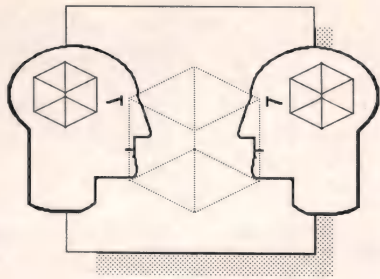
04-05-86 11:03:44 dmem.prg  
Sun 04-05-86 13:47:57

```

1 PUBLIC value, val1, val2, val3
2 USE val1val2, index, dates
3 date=now()/"12/30/85"
4 DO WHILE date() <= "01/01/86"
5   date= date() + "01/01/86"
6   IF NOT (date() <= "01/01/86")
7     value = 0.00
8     val1 = 0.00
9     val2 = 0.00
10    val3 = 0.00
11    IF NOT (date() <= "01/01/86")
12      CASE Selector = "1"
13      CASE Selector = "2"
14      CASE Selector = "3"
15      CASE Selector = "4"
16      CASE Selector = "5"
17      CASE Selector = "6"
18      CASE Selector = "7"
19      CASE Selector = "8"
20      CASE Selector = "9"
21      CASE Selector = "A"
22      CASE Selector = "B"
23      CASE Selector = "C"
24      CASE Selector = "D"
25      CASE Selector = "E"
26      CASE Selector = "F"
27      CASE Selector = "G"
28      CASE Selector = "H"
29      CASE Selector = "I"
30      CASE Selector = "J"
31      CASE Selector = "K"
32      CASE Selector = "L"
33      CASE Selector = "M"
34      CASE Selector = "N"
35      CASE Selector = "O"
36      CASE Selector = "P"
37      CASE Selector = "Q"
38      CASE Selector = "R"
39      CASE Selector = "S"
40      CASE Selector = "T"
41      CASE Selector = "U"
42      CASE Selector = "V"
43      CASE Selector = "W"
44      CASE Selector = "X"
45      CASE Selector = "Y"
46      CASE Selector = "Z"
47      CASE Selector = "0"
48      CASE Selector = "1"
49      CASE Selector = "2"
50      CASE Selector = "3"
51      CASE Selector = "4"
52      CASE Selector = "5"
53      CASE Selector = "6"
54      CASE Selector = "7"
55      CASE Selector = "8"
56      CASE Selector = "9"
57      CASE Selector = "A"
58      CASE Selector = "B"
59      CASE Selector = "C"
60      CASE Selector = "D"
61      CASE Selector = "E"
62      CASE Selector = "F"
63      CASE Selector = "G"
64      CASE Selector = "H"
65      CASE Selector = "I"
66      CASE Selector = "J"
67      CASE Selector = "K"
68      CASE Selector = "L"
69      CASE Selector = "M"
70      CASE Selector = "N"
71      CASE Selector = "O"
72      CASE Selector = "P"
73      CASE Selector = "Q"
74      CASE Selector = "R"
75      CASE Selector = "S"
76      CASE Selector = "T"
77      CASE Selector = "U"
78      CASE Selector = "V"
79      CASE Selector = "W"
80      CASE Selector = "X"
81      CASE Selector = "Y"
82      CASE Selector = "Z"
83      CASE Selector = "0"
84      CASE Selector = "1"
85      CASE Selector = "2"
86      CASE Selector = "3"
87      CASE Selector = "4"
88      CASE Selector = "5"
89      CASE Selector = "6"
90      CASE Selector = "7"
91      CASE Selector = "8"
92      CASE Selector = "9"
93      CASE Selector = "A"
94      CASE Selector = "B"
95      CASE Selector = "C"
96      CASE Selector = "D"
97      CASE Selector = "E"
98      CASE Selector = "F"
99      CASE Selector = "G"
100     CASE Selector = "H"
101     CASE Selector = "I"
102     CASE Selector = "J"
103     CASE Selector = "K"
104     CASE Selector = "L"
105     CASE Selector = "M"
106     CASE Selector = "N"
107     CASE Selector = "O"
108     CASE Selector = "P"
109     CASE Selector = "Q"
110     CASE Selector = "R"
111     CASE Selector = "S"
112     CASE Selector = "T"
113     CASE Selector = "U"
114     CASE Selector = "V"
115     CASE Selector = "W"
116     CASE Selector = "X"
117     CASE Selector = "Y"
118     CASE Selector = "Z"
119     CASE Selector = "0"
120     CASE Selector = "1"
121     CASE Selector = "2"
122     CASE Selector = "3"
123     CASE Selector = "4"
124     CASE Selector = "5"
125     CASE Selector = "6"
126     CASE Selector = "7"
127     CASE Selector = "8"
128     CASE Selector = "9"
129     CASE Selector = "A"
130     CASE Selector = "B"
131     CASE Selector = "C"
132     CASE Selector = "D"
133     CASE Selector = "E"
134     CASE Selector = "F"
135     CASE Selector = "G"
136     CASE Selector = "H"
137     CASE Selector = "I"
138     CASE Selector = "J"
139     CASE Selector = "K"
140     CASE Selector = "L"
141     CASE Selector = "M"
142     CASE Selector = "N"
143     CASE Selector = "O"
144     CASE Selector = "P"
145     CASE Selector = "Q"
146     CASE Selector = "R"
147     CASE Selector = "S"
148     CASE Selector = "T"
149     CASE Selector = "U"
150     CASE Selector = "V"
151     CASE Selector = "W"
152     CASE Selector = "X"
153     CASE Selector = "Y"
154     CASE Selector = "Z"
155     CASE Selector = "0"
156     CASE Selector = "1"
157     CASE Selector = "2"
158     CASE Selector = "3"
159     CASE Selector = "4"
160     CASE Selector = "5"
161     CASE Selector = "6"
162     CASE Selector = "7"
163     CASE Selector = "8"
164     CASE Selector = "9"
165     CASE Selector = "A"
166     CASE Selector = "B"
167     CASE Selector = "C"
168     CASE Selector = "D"
169     CASE Selector = "E"
170     CASE Selector = "F"
171     CASE Selector = "G"
172     CASE Selector = "H"
173     CASE Selector = "I"
174     CASE Selector = "J"
175     CASE Selector = "K"
176     CASE Selector = "L"
177     CASE Selector = "M"
178     CASE Selector = "N"
179     CASE Selector = "O"
180     CASE Selector = "P"
181     CASE Selector = "Q"
182     CASE Selector = "R"
183     CASE Selector = "S"
184     CASE Selector = "T"
185     CASE Selector = "U"
186     CASE Selector = "V"
187     CASE Selector = "W"
188     CASE Selector = "X"
189     CASE Selector = "Y"
190     CASE Selector = "Z"
191     CASE Selector = "0"
192     CASE Selector = "1"
193     CASE Selector = "2"
194     CASE Selector = "3"
195     CASE Selector = "4"
196     CASE Selector = "5"
197     CASE Selector = "6"
198     CASE Selector = "7"
199     CASE Selector = "8"
200     CASE Selector = "9"
201     CASE Selector = "A"
202     CASE Selector = "B"
203     CASE Selector = "C"
204     CASE Selector = "D"
205     CASE Selector = "E"
206     CASE Selector = "F"
207     CASE Selector = "G"
208     CASE Selector = "H"
209     CASE Selector = "I"
210     CASE Selector = "J"
211     CASE Selector = "K"
212     CASE Selector = "L"
213     CASE Selector = "M"
214     CASE Selector = "N"
215     CASE Selector = "O"
216     CASE Selector = "P"
217     CASE Selector = "Q"
218     CASE Selector = "R"
219     CASE Selector = "S"
220     CASE Selector = "T"
221     CASE Selector = "U"
222     CASE Selector = "V"
223     CASE Selector = "W"
224     CASE Selector = "X"
225     CASE Selector = "Y"
226     CASE Selector = "Z"
227     CASE Selector = "0"
228     CASE Selector = "1"
229     CASE Selector = "2"
230     CASE Selector = "3"
231     CASE Selector = "4"
232     CASE Selector = "5"
233     CASE Selector = "6"
234     CASE Selector = "7"
235     CASE Selector = "8"
236     CASE Selector = "9"
237     CASE Selector = "A"
238     CASE Selector = "B"
239     CASE Selector = "C"
240     CASE Selector = "D"
241     CASE Selector = "E"
242     CASE Selector = "F"
243     CASE Selector = "G"
244     CASE Selector = "H"
245     CASE Selector = "I"
246     CASE Selector = "J"
247     CASE Selector = "K"
248     CASE Selector = "L"
249     CASE Selector = "M"
250     CASE Selector = "N"
251     CASE Selector = "O"
252     CASE Selector = "P"
253     CASE Selector = "Q"
254     CASE Selector = "R"
255     CASE Selector = "S"
256     CASE Selector = "T"
257     CASE Selector = "U"
258     CASE Selector = "V"
259     CASE Selector = "W"
260     CASE Selector = "X"
261     CASE Selector = "Y"
262     CASE Selector = "Z"
263     CASE Selector = "0"
264     CASE Selector = "1"
265     CASE Selector = "2"
266     CASE Selector = "3"
267     CASE Selector = "4"
268     CASE Selector = "5"
269     CASE Selector = "6"
270     CASE Selector = "7"
271     CASE Selector = "8"
272     CASE Selector = "9"
273     CASE Selector = "A"
274     CASE Selector = "B"
275     CASE Selector = "C"
276     CASE Selector = "D"
277     CASE Selector = "E"
278     CASE Selector = "F"
279     CASE Selector = "G"
280     CASE Selector = "H"
281     CASE Selector = "I"
282     CASE Selector = "J"
283     CASE Selector = "K"
284     CASE Selector = "L"
285     CASE Selector = "M"
286     CASE Selector = "N"
287     CASE Selector = "O"
288     CASE Selector = "P"
289     CASE Selector = "Q"
290     CASE Selector = "R"
291     CASE Selector = "S"
292     CASE Selector = "T"
293     CASE Selector = "U"
294     CASE Selector = "V"
295     CASE Selector = "W"
296     CASE Selector = "X"
297     CASE Selector = "Y"
298     CASE Selector = "Z"
299     CASE Selector = "0"
300     CASE Selector = "1"
301     CASE Selector = "2"
302     CASE Selector = "3"
303     CASE Selector = "4"
304     CASE Selector = "5"
305     CASE Selector = "6"
306     CASE Selector = "7"
307     CASE Selector = "8"
308     CASE Selector = "9"
309     CASE Selector = "A"
310     CASE Selector = "B"
311     CASE Selector = "C"
312     CASE Selector = "D"
313     CASE Selector = "E"
314     CASE Selector = "F"
315     CASE Selector = "G"
316     CASE Selector = "H"
317     CASE Selector = "I"
318     CASE Selector = "J"
319     CASE Selector = "K"
320     CASE Selector = "L"
321     CASE Selector = "M"
322     CASE Selector = "N"
323     CASE Selector = "O"
324     CASE Selector = "P"
325     CASE Selector = "Q"
326     CASE Selector = "R"
327     CASE Selector = "S"
328     CASE Selector = "T"
329     CASE Selector = "U"
330     CASE Selector = "V"
331     CASE Selector = "W"
332     CASE Selector = "X"
333     CASE Selector = "Y"
334     CASE Selector = "Z"
335     CASE Selector = "0"
336     CASE Selector = "1"
337     CASE Selector = "2"
338     CASE Selector = "3"
339     CASE Selector = "4"
340     CASE Selector = "5"
341     CASE Selector = "6"
342     CASE Selector = "7"
343     CASE Selector = "8"
344     CASE Selector = "9"
345     CASE Selector = "A"
346     CASE Selector = "B"
347     CASE Selector = "C"
348     CASE Selector = "D"
349     CASE Selector = "E"
350     CASE Selector = "F"
351     CASE Selector = "G"
352     CASE Selector = "H"
353     CASE Selector = "I"
354     CASE Selector = "J"
355     CASE Selector = "K"
356     CASE Selector = "L"
357     CASE Selector = "M"
358     CASE Selector = "N"
359     CASE Selector = "O"
360     CASE Selector = "P"
361     CASE Selector = "Q"
362     CASE Selector = "R"
363     CASE Selector = "S"
364     CASE Selector = "T"
365     CASE Selector = "U"
366     CASE Selector = "V"
367     CASE Selector = "W"
368     CASE Selector = "X"
369     CASE Selector = "Y"
370     CASE Selector = "Z"
371     CASE Selector = "0"
372     CASE Selector = "1"
373     CASE Selector = "2"
374     CASE Selector = "3"
375     CASE Selector = "4"
376     CASE Selector = "5"
377     CASE Selector = "6"
378     CASE Selector = "7"
379     CASE Selector = "8"
380     CASE Selector = "9"
381     CASE Selector = "A"
382     CASE Selector = "B"
383     CASE Selector = "C"
384     CASE Selector = "D"
385     CASE Selector = "E"
386     CASE Selector = "F"
387     CASE Selector = "G"
388     CASE Selector = "H"
389     CASE Selector = "I"
390     CASE Selector = "J"
391     CASE Selector = "K"
392     CASE Selector = "L"
393     CASE Selector = "M"
394     CASE Selector = "N"
395     CASE Selector = "O"
396     CASE Selector = "P"
397     CASE Selector = "Q"
398     CASE Selector = "R"
399     CASE Selector = "S"
400     CASE Selector = "T"
401     CASE Selector = "U"
402     CASE Selector = "V"
403     CASE Selector = "W"
404     CASE Selector = "X"
405     CASE Selector = "Y"
406     CASE Selector = "Z"
407     CASE Selector = "0"
408     CASE Selector = "1"
409     CASE Selector = "2"
410     CASE Selector = "3"
411     CASE Selector = "4"
412     CASE Selector = "5"
413     CASE Selector = "6"
414     CASE Selector = "7"
415     CASE Selector = "8"
416     CASE Selector = "9"
417     CASE Selector = "A"
418     CASE Selector = "B"
419     CASE Selector = "C"
420     CASE Selector = "D"
421     CASE Selector = "E"
422     CASE Selector = "F"
423     CASE Selector = "G"
424     CASE Selector = "H"
425     CASE Selector = "I"
426     CASE Selector = "J"
427     CASE Selector = "K"
428     CASE Selector = "L"
429     CASE Selector = "M"
430     CASE Selector = "N"
431     CASE Selector = "O"
432     CASE Selector = "P"
433     CASE Selector = "Q"
434     CASE Selector = "R"
435     CASE Selector = "S"
436     CASE Selector = "T"
437     CASE Selector = "U"
438     CASE Selector = "V"
439     CASE Selector = "W"
440     CASE Selector = "X"
441     CASE Selector = "Y"
442     CASE Selector = "Z"
443     CASE Selector = "0"
444     CASE Selector = "1"
445     CASE Selector = "2"
446     CASE Selector = "3"
447     CASE Selector = "4"
448     CASE Selector = "5"
449     CASE Selector = "6"
450     CASE Selector = "7"
451     CASE Selector = "8"
452     CASE Selector = "9"
453     CASE Selector = "A"
454     CASE Selector = "B"
455     CASE Selector = "C"
456     CASE Selector = "D"
457     CASE Selector = "E"
458     CASE Selector = "F"
459     CASE Selector = "G"
460     CASE Selector = "H"
461     CASE Selector = "I"
462     CASE Selector = "J"
463     CASE Selector = "K"
464     CASE Selector = "L"
465     CASE Selector = "M"
466     CASE Selector = "N"
467     CASE Selector = "O"
468     CASE Selector = "P"
469     CASE Selector = "Q"
470     CASE Selector = "R"
471     CASE Selector = "S"
472     CASE Selector = "T"
473     CASE Selector = "U"
474     CASE Selector = "V"
475     CASE Selector = "W"
476     CASE Selector = "X"
477     CASE Selector = "Y"
478     CASE Selector = "Z"
479     CASE Selector = "0"
480     CASE Selector = "1"
481     CASE Selector = "2"
482     CASE Selector = "3"
483     CASE Selector = "4"
484     CASE Selector = "5"
485     CASE Selector = "6"
486     CASE Selector = "7"
487     CASE Selector = "8"
488     CASE Selector = "9"
489     CASE Selector = "A"
490     CASE Selector = "B"
491     CASE Selector = "C"
492     CASE Selector = "D"
493     CASE Selector = "E"
494     CASE Selector = "F"
495     CASE Selector = "G"
496     CASE Selector = "H"
497     CASE Selector = "I"
498     CASE Selector = "J"
499     CASE Selector = "K"
500     CASE Selector = "L"
501     CASE Selector = "M"
502     CASE Selector = "N"
503     CASE Selector = "O"
504     CASE Selector = "P"
505     CASE Selector = "Q"
506     CASE Selector = "R"
507     CASE Selector = "S"
508     CASE Selector = "T"
509     CASE Selector = "U"
510     CASE Selector = "V"
511     CASE Selector = "W"
512     CASE Selector = "X"
513     CASE Selector = "Y"
514     CASE Selector = "Z"
515     CASE Selector = "0"
516     CASE Selector = "1"
517     CASE Selector = "2"
518     CASE Selector = "3"
519     CASE Selector = "4"
520     CASE Selector = "5"
521     CASE Selector = "6"
522     CASE Selector = "7"
523     CASE Selector = "8"
524     CASE Selector = "9"
525     CASE Selector = "A"
526     CASE Selector = "B"
527     CASE Selector = "C"
528     CASE Selector = "D"
529     CASE Selector = "E"
530     CASE Selector = "F"
531     CASE Selector = "G"
532     CASE Selector = "H"
533     CASE Selector = "I"
534     CASE Selector = "J"
535     CASE Selector = "K"
536     CASE Selector = "L"
537     CASE Selector = "M"
538     CASE Selector = "N"
539     CASE Selector = "O"
540     CASE Selector = "P"
541     CASE Selector = "Q"
542     CASE Selector = "R"
543     CASE Selector = "S"
544     CASE Selector = "T"
545     CASE Selector = "U"
546     CASE Selector = "V"
547     CASE Selector = "W"
548     CASE Selector = "X"
549     CASE Selector = "Y"
550     CASE Selector = "Z"
551     CASE Selector = "0"
552     CASE Selector = "1"
553     CASE Selector = "2"
554     CASE Selector = "3"
555     CASE Selector = "4"
556     CASE Selector = "5"
557     CASE Selector = "6"
558     CASE Selector = "7"
559     CASE Selector = "8"
560     CASE Selector = "9"
561     CASE Selector = "A"
562     CASE Selector = "B"
563     CASE Selector = "C"
564     CASE Selector = "D"
565     CASE Selector = "E"
566     CASE Selector = "F"
567     CASE Selector = "G"
568     CASE Selector = "H"
569     CASE Selector = "I"
570     CASE Selector = "J"
571     CASE Selector = "K"
572     CASE Selector = "L"
573     CASE Selector = "M"
574     CASE Selector = "N"
575     CASE Selector = "O"
576     CASE Selector = "P"
577     CASE Selector = "Q"
578     CASE Selector = "R"
579     CASE Selector = "S"
580     CASE Selector = "T"
581     CASE Selector = "U"
582     CASE Selector = "V"
583     CASE Selector = "W"
584     CASE Selector = "X"
585     CASE Selector = "Y"
586     CASE Selector = "Z"
587     CASE Selector = "0"
588     CASE Selector = "1"
589     CASE Selector = "2"
590     CASE Selector = "3"
591     CASE Selector = "4"
592     CASE Selector = "5"
593     CASE Selector = "6"
594     CASE Selector = "7"
595     CASE Selector = "8"
596     CASE Selector = "9"
597     CASE Selector = "A"
598     CASE Selector = "B"
599     CASE Selector = "C"
600     CASE Selector = "D"
601     CASE Selector = "E"
602     CASE Selector = "F"
603     CASE Selector = "G"
604     CASE Selector = "H"
605     CASE Selector = "I"
606     CASE Selector = "J"
607     CASE Selector = "K"
608     CASE Selector = "L"
609     CASE Selector = "M"
610     CASE Selector = "N"
611     CASE Selector = "O"
612     CASE Selector = "P"
613     CASE Selector = "Q"
614     CASE Selector = "R"
615     CASE Selector = "S"
616     CASE Selector = "T"
617     CASE Selector = "U"
618     CASE Selector = "V"
619     CASE Selector = "W"
620     CASE Selector = "X"
621     CASE Selector = "Y"
622     CASE Selector = "Z"
623     CASE Selector = "0"
624     CASE Selector = "1"
625     CASE Selector = "2"
626     CASE Selector = "3"
627     CASE Selector = "4"
628     CASE Selector = "5"
629     CASE Selector = "6"
630     CASE Selector = "7"
631     CASE Selector = "8"
632     CASE Selector = "9"
633     CASE Selector = "A"
634     CASE Selector = "B"
635     CASE Selector = "C"
636     CASE Selector = "D"
637     CASE Selector = "E"
638     CASE Selector = "F"
639     CASE Selector = "G"
640     CASE Selector = "H"
641     CASE Selector = "I"
642     CASE Selector = "J"
643     CASE Selector = "K"
644     CASE Selector = "L"
645     CASE Selector = "M"
646     CASE Selector = "N"
647     CASE Selector = "O"
648     CASE Selector = "P"
649     CASE Selector = "Q"
650     CASE Selector = "R"
651     CASE Selector = "S"
652     CASE Selector = "T"
653     CASE Selector = "U"
654     CASE Selector = "V"
655     CASE Selector = "W"
656     CASE Selector = "X"
657     CASE Selector = "Y"
658     CASE Selector = "Z"
659     CASE Selector = "0"
660     CASE Selector = "1"
661     CASE Selector = "2"
662     CASE Selector = "3"
663     CASE Selector = "4"
664     CASE Selector = "5"
665     CASE Selector = "6"
666     CASE Selector = "7"
667     CASE Selector = "8"
668     CASE Selector = "9"
669     CASE Selector = "A"
670     CASE Selector = "B"
671     CASE Selector = "C"
672     CASE Selector = "D"
673     CASE Selector = "E"
674     CASE Selector = "F"
675     CASE Selector = "G"
676     CASE Selector = "H"
677     CASE Selector = "I"
678     CASE Selector = "J"
679     CASE Selector = "K"
680     CASE Selector = "L"
681     CASE Selector = "M"
682     CASE Selector = "N"
683     CASE Selector = "O"
684     CASE Selector = "P"
685     CASE Selector = "Q"
686     CASE Selector = "R"
687     CASE Selector = "S"
688     CASE Selector = "T"
689     CASE Selector = "U"
690     CASE Selector = "V"
691     CASE Selector = "W"
692     CASE Selector = "X"
693     CASE Selector = "Y"
694     CASE Selector = "Z"
695     CASE Selector = "0"
696     CASE Selector = "1"
697     CASE Selector = "2"
698     CASE Selector = "3"
699     CASE Selector = "4"
700     CASE Selector = "5"
701     CASE Selector = "6"
702     CASE Selector = "7"
703     CASE Selector = "8"
704     CASE Selector = "9"
705     CASE Selector = "A"
706     CASE Selector = "B"
707     CASE Selector = "C"
708     CASE Selector = "D"
709     CASE Selector = "E"
710     CASE Selector = "F"
711     CASE Selector = "G"
712     CASE Selector = "H"
713     CASE Selector = "I"
714     CASE Selector = "J"
715     CASE Selector = "K"
716     CASE Selector = "L"
717     CASE Selector = "M"
718     CASE Selector = "N"
719     CASE Selector = "O"
720     CASE Selector = "P"
721     CASE Selector = "Q"
722     CASE Selector = "R"
723     CASE Selector = "S"
724     CASE Selector = "T"
725     CASE Selector = "U"
726     CASE Selector = "V"
727     CASE Selector = "W"
728     CASE Selector = "X"
729     CASE Selector = "Y"
730     CASE Selector = "Z"
731     CASE Selector = "0"
732     CASE Selector = "1"
733     CASE Selector = "2"
734     CASE Selector = "3"
735     CASE Selector = "4"
736     CASE Selector = "5"
737     CASE Selector = "6"
738     CASE Selector = "7"
739     CASE Selector = "8"
740     CASE Selector = "9"
741     CASE Selector = "A"
742     CASE Selector = "B"
743     CASE Selector = "C"
744     CASE Selector = "D"
745     CASE Selector = "E"
746     CASE Selector = "F"
747     CASE Selector = "G"
748     CASE Selector = "H"
749     CASE Selector = "I"
750     CASE Selector = "J"
751     CASE Selector = "K"
752     CASE Selector = "L"
753     CASE Selector = "M"
754     CASE Selector = "N"
755     CASE Selector = "O"
756     CASE Selector = "P"
757     CASE Selector = "Q"
758     CASE Selector = "R"
759     CASE Selector = "S"
760     CASE Selector = "T"
761     CASE Selector = "U"
762     CASE Selector = "V"
763     CASE Selector = "W"
764     CASE Selector = "X"
765     CASE Selector = "Y"
766     CASE Selector = "Z"
767     CASE Selector = "0"
768     CASE Selector = "1"
769     CASE Selector = "2"
770     CASE Selector = "3"
771     CASE Selector = "4"
772     CASE Selector = "5"
773     CASE Selector = "6"
774     CASE Selector = "7"
775     CASE Selector = "8"
776     CASE Selector = "9"
777     CASE Selector = "A"
778     CASE Selector = "B"
779     CASE Selector = "C"
780     CASE Selector = "D"
781     CASE Selector = "E"
782     CASE Selector = "F"
783     CASE Selector = "G"
784     CASE Selector = "H"
785     CASE Selector = "I"
786     CASE Selector = "J"
787     CASE Selector = "K"
788     CASE Selector = "L"
789     CASE Selector = "M"
790     CASE Selector = "N"
791     CASE Selector = "O"
792     CASE Selector = "P"
793     CASE Selector = "Q"
794     CASE Selector = "R"
795     CASE Selector = "S"
796     CASE Selector = "T"
797     CASE Selector = "U"
798     CASE Selector = "V"
799     CASE Selector = "W"
800     CASE Selector = "X"
801     CASE Selector = "Y"
802     CASE Selector = "Z"
803     CASE Selector = "0"
804     CASE Selector = "1"
805     CASE Selector = "2"
806     CASE Selector = "3"
807     CASE Selector = "4"
808     CASE Selector = "5"
809     CASE Selector = "6"
810     CASE Selector = "7"
811     CASE Selector = "8"
812     CASE Selector = "9"
813     CASE Selector = "A"
814     CASE Selector = "B"
815     CASE Selector = "C"
816     CASE Selector = "D"
817     CASE Selector = "E"
818     CASE Selector = "F"
819     CASE Selector = "G"
820     CASE Selector = "H"
821     CASE Selector = "I"
822     CASE Selector = "J"
823     CASE Selector = "K"
824     CASE Selector = "L"
825     CASE Selector = "M"
826     CASE Selector = "N"
827     CASE Selector = "O"
828     CASE Selector = "P"
829     CASE Selector = "Q"
830     CASE Selector = "R"
831     CASE Selector = "S"
832     CASE Selector = "T"
833     CASE Selector = "U"
834     CASE Selector = "V"
835     CASE Selector = "W"
836     CASE Selector = "X"
837     CASE Selector = "Y"
838     CASE Selector = "Z"
839     CASE Selector = "0"
840     CASE Selector = "1"
841     CASE Selector = "2"
842     CASE Selector = "3"
843     CASE Selector = "4"
844     CASE Selector = "5"
845     CASE Selector = "6"
846     CASE Selector = "7"
847     CASE Selector = "8"
848     CASE Selector = "9"
849     CASE Selector = "A"
850     CASE Selector = "B"
851     CASE Selector = "C"
852     CASE Selector = "D"
853     CASE Selector = "E"
854     CASE Selector = "F"
855     CASE Selector = "G"
856     CASE Selector = "H"
857     CASE Selector = "I"
858     CASE Selector = "J"
859     CASE Selector = "K"
860     CASE Selector = "L"
861     CASE Selector = "M"
862     CASE Selector = "N"
863     CASE Selector = "O"
864     CASE Selector = "P"
865     CASE Selector = "Q"
866     CASE Selector = "R"
867     CASE Selector = "S"
868     CASE Selector = "T"
869     CASE Selector = "U"
870     CASE Selector = "V"
871     CASE Selector = "W"
872     CASE Selector = "X"
873     CASE Selector = "Y"
874     CASE Selector = "Z"
875     CASE Selector = "0"
876     CASE Selector = "1"
877     CASE Selector = "2"
878     CASE Selector = "3"
879     CASE Selector = "4"
880     CASE Selector = "5"
881     CASE Selector = "6"
882     CASE Selector = "7"
883     CASE Selector = "8"
884     CASE Selector = "9"
885     CASE Selector = "A"
886     CASE Selector = "
```



# VIEWPOINT



## What's Right with High-Level Languages?

In "What's Wrong with High-Level Languages" (Viewpoint, February 1987), Mike Suman brings up some interesting points about the limitations inherent in any computer language but fails to mention any of the strengths of Modula-2 or of modern high-level languages in general.

As a college instructor who has spent several years developing and teaching a course in assembly-language programming, I can well appreciate the advantages of low-level programming. Assembly language has some shortcomings that can be much more serious than the limitations of high-level languages, however.

Assembly language is much harder to learn than are most high-level languages, and you must substantially relearn it if you move from one computer to another. I grant that the second assembly language is much easier to learn than the first. It would, however, take a considerable effort to adapt from the 68000 to the

are the conditional and unconditional jump and the call to subroutine. To test several conditions for a loop, you must write a cascade of comparisons and jumps. To produce a simple repetition, you must split the control instructions between the outside of the loop, the top of the loop, and the bottom of the loop. Consider Examples 1 and 2, below, in which some elements of a character array are cleared using 68000 assembly language and Modula-2, respectively. In both examples, the range of items to be cleared has been calculated and left in variables named *TOP* and *BOT*. In the assembly-language example, the loop constraints are developed in five different lines of code, whereas

in the high-level language example, all loop constraints are developed on a single line.

Modern high-level languages allow programmers to express data in terms that naturally match a wide range of typical problems. Modula-2 handles the mathematical concepts of real and integral numeric types and sets, the organizational concepts of records and files, and the universal concepts of arrays and characters. In assembly language, the only data type is the computer word. Programmers must impose a structure and then provide algorithms to perform even the simplest task. Consider how much easier it is (for exam-

*(continued on page 124)*

```

*****
*   FOR loop in 68000
*
*   First Array Element to Process: TOP
*   Last Array Element to Process: BOT
*   The array: DATA
*
*   Used to Index into array: DO
*   Used as Pointer to array: AO
*
*****
DATA    DS    500           ;array of bytes
TOP     DC    0             ;first to process
BOT     DC    0             ;last to process
*
*                                     ;other data
*                                     ;and code
*
*
*   Assumes TOP & BOT previously calculated
*
0001FB  41F900000000        LEA    DATA,AO      ;set pointer
0001FE  3039000001F4        MOVE   TOP,DO        ;init index
000204  B079000001F6        CMP    BOT,DO        ;check bounds
00020A  6206                BHI,S   LABEL        ;end loop
00020C  42300000            CLR.B   0(AO,DO)      ;loop body
000210  60F2                BRA.S   LOOP          ;repeat

                                LABEL
000212                                END

```

**Example 1:** Clearing elements of a character array in 68000 assembly language

```

MODULE TRIAL;

VAR
  DATA : ARRAY [1..500] OF CHAR;
  TOP, BOT, i : CARDINAL;
BEGIN
  (* Assumes TOP & BOT have been previously calculated *)

  FOR i := TOP TO BOT DO
    DATA[i] := 0;
  END;
END TRIAL.

```

**Example 2:** Modula-2 version of the code in Example 1

by Brian R. Anderson

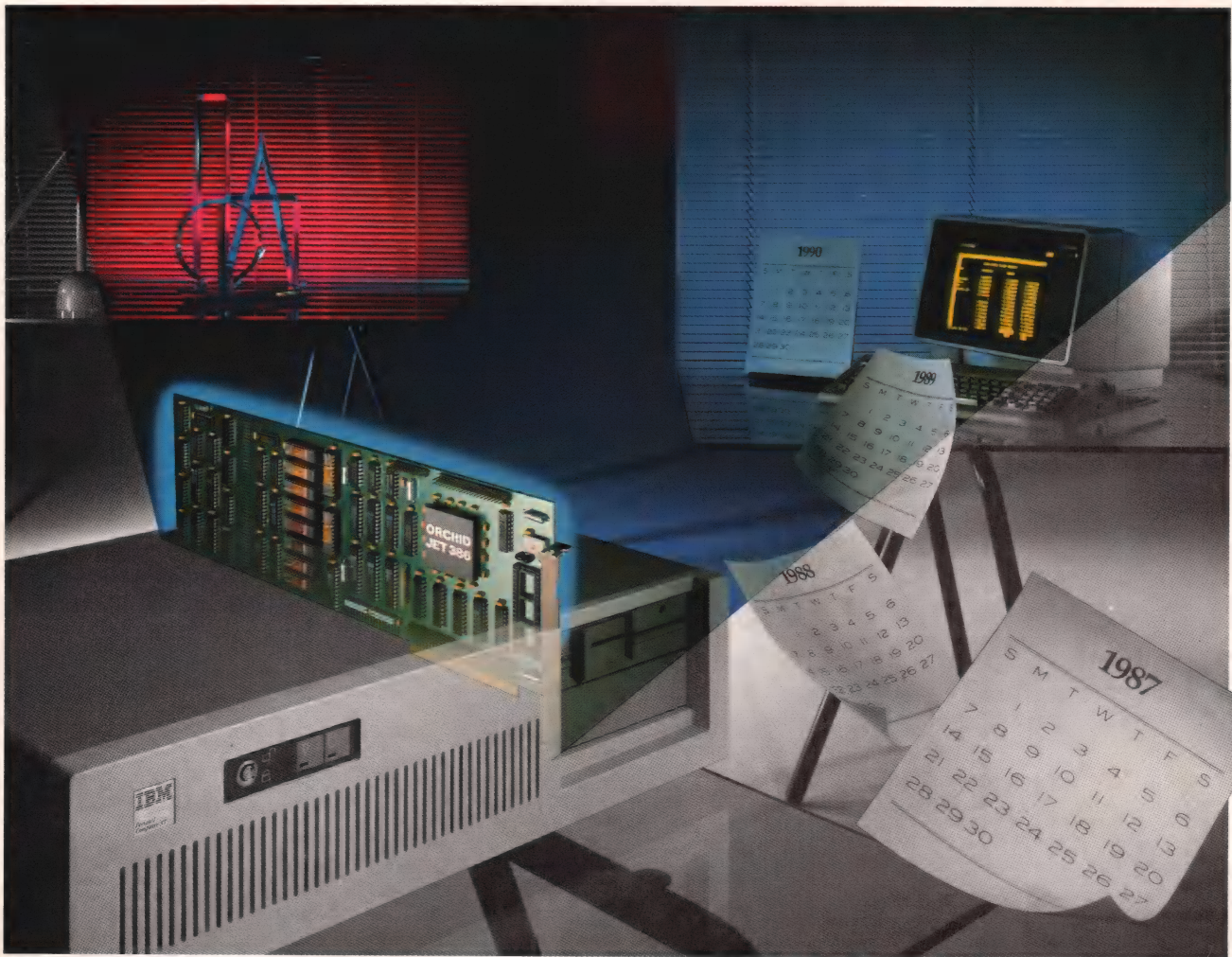
8086, for instance, especially when you consider the peculiarities of Microsoft's MASM.

Assembly language is hard to write and even harder to debug (compared to Modula-2), partially because often the only control structures available

Brian R. Anderson, 5105 Lorraine Ave., Burnaby, B.C. V5G 2S3 Canada. Brian is an instructor in the Electronics Dept. at the Vancouver Vocational Institute.



# ORCHID'S JET 386™: POWER FOR THE FUTURE NOW



## Jet 386 is the Ultimate Accelerator Upgrade for Your AT

Announcing an end to obsolescence. Orchid Technology's Jet 386™ accelerator card extends the life of your computer investment into the 1990s—it puts power in your AT that you won't outgrow.

### Three Times Faster than an AT

It's up to three times faster than an AT depending on the application, and speed is just one benefit. Unequalled compatibility and provisions for upcoming 386 software mean your Jet 386 will handle whatever the future has in store: CAD, spreadsheets, networking...

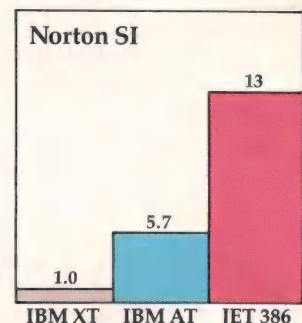
### Easy Upgrade

Easy to use, there's nothing new to learn and no new programs to buy. At 25% of the cost of buying a new 386 PC, it's easy on your pocketbook, too.

### From the People Who Started It All

Orchid combined 80386 power with the technology perfected for the XT in the TinyTurbo and PCturbo 286e. Like these critically acclaimed accelerators, Jet 386 is built for lasting value.

Call Orchid to find out how you can experience the future today. And ask how Orchid can modernize your whole office with turbos, graphics, networking, and multifunction products.



Jet 386, PCturbo 286e and TinyTurbo 286 are trademarks of Orchid Technology. All other products named are trademarks of their manufacturers.

**CIRCLE 130 ON READER SERVICE CARD**

45365 Northport Loop West  
Fremont, CA 94538  
415/490-8586 Tlx: 709289



# An Efficient Algorithm for Large Priority Queues

by Robert Jay Brown

In many real-time programs, it is necessary to share resources that are in short supply. To help manage these situations, a priority system is often established. When several contenders are competing for the same resource, the one with the highest priority gets the resource. When the resource again becomes available, the next-priority contender gets it. A simple first-in/first-out queue can be thought of as a priority queue in which the time a contender is enqueued is the priority and lower numbers have higher priority.

In practice, the resource being waited on may be a physical device, such as a printer; a logical device, such as a file or a record of a file; or the CPU itself. Alternatively, a timer scheduler may be implemented by using the desired dispatch time as the priority and having the de-queueing operation wait until the time of day equals the dispatch time at the head of the queue.

## Basic Queue Operations

A priority queueing scheme in a real-time system must be

*The central concept is that inserting and removing from the queue can be viewed as merging two separate queues.*

able to perform the following operations on the elements, or nodes, of the queue: determine the highest-priority node, add a new node, and remove a node. Removing the highest-priority node is a special case of the more general operation

of removing a node anywhere in the queue. This is called preempting.

The most simple implementation for priority queues results in the time to perform at least one of the above operations being directly proportional to the size of the queue. For extremely large queues, this becomes unworkable. The problem is similar to sorting, and sorting can be done in a time proportional to the logarithm of the number of elements, so you should be able to do as well for a priority queue.

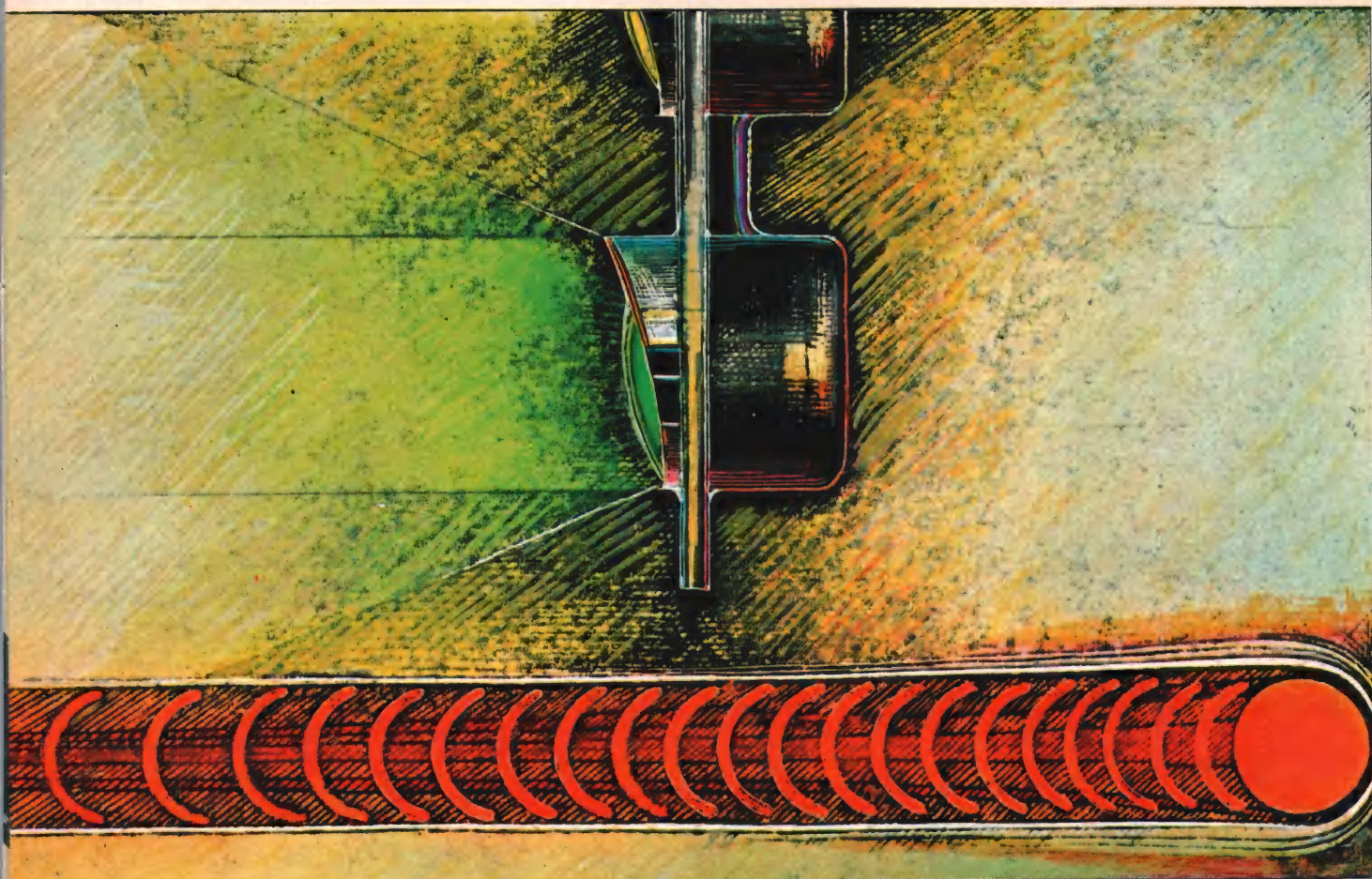
In fact, you can do a bit better than this. Although the priority queue algorithm I have implemented (see Example 1, page 18) performs in logarithmic time, it does not keep the queue completely sorted. It defers determining the second-priority element until after the top-priority element is removed. This does not change the logarithmic component, but it makes each iteration go faster.

## A Binary Tree

The representation for the queue is a binary tree with left, right, and father connections. Each element also con-

Robert Jay Brown, 301 W. High St., P.O. Box 833, Warsaw, KY 41095. Robert is a graduate student at Florida Atlantic University. He is a consultant involved in designing electronic surveillance intercept and cryptography systems.





tains a sort *key*, which can be interpreted as the priority. In addition, a *dist* cell is used to indicate the minimum distance from that node to a leaf, which is a connection to no element, or a terminator. The *father* connection is not used to maintain the ordering of the queue but is used to allow rapid removal of any node in the queue (see the example, lines 28–37). (Note: my structure declaring words [lines 3–26], together with examples of their use, are available on the East Coast Forth Board as the file STRUC.ARC, and so I will not describe them further here.)

The *word* and *data* cells (lines 36–37) are used to contain dispatching information. *Word* is the address of a Forth word to *perform* when the node is dispatched, and *data* is a word of data, typically a pointer, that is pushed on the stack before *performing* the *word*.

The following priority queueing algorithm was first described by Clark Allen Crane in 1971. My implementation is an extension of a revised version of Crane's algorithm that is described by Donald Knuth. You can refer to Knuth's *The Art of Computer Programming: Volume 3, Sorting and Searching* (Reading, Mass.: Addison-Wesley, 1973) for a complete description of the algorithm and an analysis of its performance.

### Merging Queues

The central concept behind Crane's algorithm is that the operations of inserting and removing an element from the queue can be viewed as merging two separate queues. Crane's algorithm keeps the highest-priority node at the root of the tree, and the subtrees follow the

same pattern. Thus, to dispatch the head element of the queue (lines 103–106), the left and right subtrees are pruned from the root and merged back together, leaving the root out of the result. To insert an element on the queue (lines 98–101), that element is viewed as a little priority queue, in its own right, of one element. This queue is merged with the original queue, and the element is thereby inserted into the original queue. To remove a node from somewhere in the middle of the queue—that is, to preempt it (lines 108–111), the node's *left* and *rite* subtrees are cut off and merged, forming an intermediate result. Next, the preempted node is removed by using its *father* pointer to cut it off from the element that points to it. Finally, the original queue, less the preempted node and its two subtrees, is merged with the intermediate tree described above.

The implementation in the example has been tested on a 10-MHz 80286 with one wait state on memory access and 3.2 percent DRAM refresh interference running under LMI PC/FORTH + 3.1. For 10,000 iterations of an *unqueue* on a randomly selected element of the queue, followed by an *enqueue* using a randomly chosen priority, it produced the results shown in Table 1, page 18.

### Availability

The complete source code, as a standard Forth screen file, including the benchmark test, is available on CompuServe (GODDJ), and on the East Coast Forth Board ([703] 442-8695). Also, all the source code for articles in this issue is available on a single disk. To order, send \$14.95 to Dr.



# DAN BRICKLIN'S DEMO PROGRAM ONLY \$74.95

Read what they're saying about this popular program for prototyping and demo-making:

**"A winner right out of the starting gate. After you use DEMO once, you'll wonder how you got along without it."**

—PC Magazine

**"Everybody who writes software, either commercially or for in-house applications, should immediately order a copy. Period. No exceptions."**

—Soft letter

## Product of the Month

—PC Tech Journal

Thousands of developers and most of the largest and best known software companies are using this program. You can, too. Act now!

# NEW TUTORIAL! JUST \$49.95

The perfect companion to the Demo Program. The Tutorial helps you learn the ins and outs of its basic and advanced features. Complete with a 96 page manual containing step-by-step instructions, diskette, and function key template.

# ORDER NOW!

1-800-CALL-800 x8088

Use 800-number for orders only.

Questions, special shipping, etc., call 617-332-2240.

No Purchase Orders. Massachusetts residents add 5% sales tax. Outside of the U.S.A., add \$15.00.

Requires 256K IBM PC/Compatible, DOS 2.0 or later.

Supports Monochrome, Color Graphics, and EGA Adapters (text mode only). The Tutorial requires the Demo Program.



## SOFTWARE GARDEN, INC.

Dept. D

P.O. Box 373, Newton Highlands, MA 02161  
CIRCLE 314 ON READER SERVICE CARD

## QUEUEING ALGORITHM

(continued from page 17)

Dobb's Journal, 501 Galveston Dr.,  
Redwood City, CA 94063 or call (415)  
366-3600 ext. 216. Please specify the  
issue number and format (MS-DOS,  
Macintosh, Kaypro).

DDJ

Vote for your favorite feature/article.  
Circle Reader Service No. 2.

#nodes	msecs
10	7.4
20	9.3
50	11.9
100	13.8
200	15.5
500	17.9
1000	19.4
2000	20.7
5000	21.9
10000	23.1

**Table 1: Benchmark test results**

```

1 ( Timer queue implementation                                rjb 16:06 07/2186 )
2
3 ( ----- structure declaring words ----- )
4
5 : <struc> CREATE , DOES> @ + ; ( 2nd generation defining word )
6
7 ( 'struc' is used to create the defining word for a structure )
8 : struc CREATE ,      ( org struc <struc> ; creates a structure )
9   DOES>      ( size <struc> <element> ; creates an element )
10  DUP @ DUP >R ROT + SWAP !   ( update location counter )
11  R> <struc> ;                ( make word for element )
12
13 ( 'array' is used to create an array of structures )
14 : array CREATE ( #slots slot-size array <array> ; makes array )
15  DUP , * ALLOT      ( save slot-size and allocate array )
16  DOES>              ( subscript <array> -- &<array>element )
17  DUP @ ROT * + WSIZE + ;   ( return pointer to the slot )
18
19 ( 'org' is useful for doing the 4th version of a C 'union' )
20 : org      ( n org <strucname> ; re-initializes location counter )
21  BL WORD FIND NOT ABORT" Undefined" >BODY ! ;
22
23 ( 'sizeof' is used for declaring structures of structures )
24 : sizeof ( sizeof <strucname> -- n ; gets the size of a struc )
25  BL WORD FIND NOT ABORT" Undefined" >BODY @
26  STATE @ IF [COMPILE] LITERAL THEN ; IMMEDIATE
27
28 ( ----- timer queue entry ----- )
29
30  0   struc   tq      ( a node in the timer queue )
31      4       tq      key   ( the sort key: dispatch time )
32      4       tq      dist  ( distance to nearest leaf )
33      4       tq      left  ( pointer to left sub-tree )
34      4       tq      rite  ( pointer to right sub-tree )
35      4       tq      father ( pointer to father of node )
36      4       tq      word  ( word to execute at dispatch )
37      4       tq      data  ( pointer to data for word )
38
39 ( ----- timer queue implementation ----- )
40
41 ( left! & rite! set the left and rite subtrees, respectively
42   of a node, called the father. The father pointer of the son
43   node is updated to point to the pointer, left or rite, that
44   points to the son, so that it may be cleared on an unque. )
45
46 : left! DUP 0<> IF OVER left OVER father !   ( Father Son -- )

```

**Example 1: Timer queue implementation**



# 2 **Programmers & Developers** **New Products!**

## **Distribute Your Demos with No Royalties**

**Screen Machine** creates interactive demos, tutorials, menu systems and DOS shells. Includes a text screen editor that optionally generates source code\* and binary or text files. Never write code for screen display again. Capture any program's text screens for editing and your own use. Capture CGA compatible graphics screens for BLOAD or direct display. SAVE hundreds of HOURS of work.

Now there's no need for separate screen and demo software packages and no need to pay outrageous royalties. Priced at only \$79.00.

\*Turbo Pascal, Mach 2 for Turbo, Assembler, dBASE II & III, BASIC (including The Inside Track and Mach 2).

## **Supercharge Turbo Pascal**

**Mach 2 for Turbo Pascal** adds assembler speed to your programs. 90 + subroutines, most in assembler, give you speed and functionality you never knew was possible. No knowledge of assembler language required.

**INSTANT** displays. **INSTANT** windows (incl. exploding and boxed). **FASTEST** sort you've seen. Read/write files **FAST** as DOS. **INSTANT** menus, 1-2-3 horizontal and vertical bar.

Trap ^C/^Break & DOS critical errors so no more A)bort, R)etry or I)gnore. Emulate **BASIC PRINT USING** for FAST formatted numbers. Execute any prog, batch or DOS command without ending program.

Read environment. Read file directory. Get/set file attributes. Plus too many string functions to describe here. No royalties when you distribute COM programs. All source code included. A true bargain at \$69.00.

NOT COPY PROTECTED. 30 Day Money-Back Performance Guarantee. Requires IBM/compatible & DOS 2+.

## **Order Now 800-922-3383**

We welcome VISA/MC. COD US only \$3. S/H US \$3, Canada \$5, Elsewhere \$18. GA res. add tax and call 404-973-9272. Demo available. Send \$5 check. Refunded on direct purchase.

We also publish Stay-Res, Mach 2 for BASIC, The Inside Track and Peeks 'n Pokes.

**MicroHelp, Inc.**  
2220 Carlyle Drive  
Marietta GA 30062

CIRCLE 215 ON READER SERVICE CARD

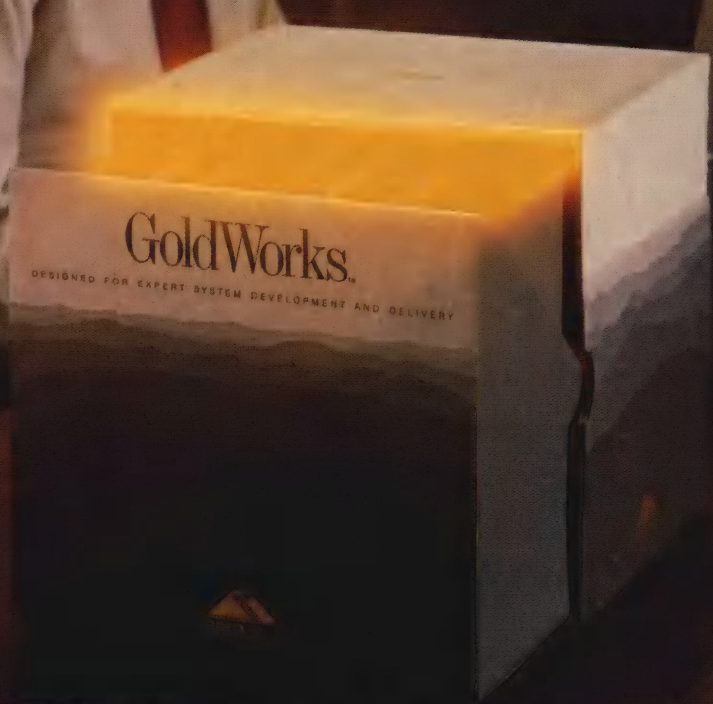
```

47      THEN SWAP left ! ;
48
49 : rite! DUP 0<> IF OVER rite OVER father !      ( Father Son -- )
50      THEN SWAP rite ! ;
51
52 : go-rt DUP rite @ >R DUP ROT rite! R> ;      ( go dn rite side )
53
54 : dist@ DUP IF dist @ THEN ;      ( P -- P=nil ? 0 : dist @ )
55
56 ( This is step M2: from Knuth p 619, sol'n to prob. 32, p 159 )
57 : list-merge      ( P Q R -- P Q R D ; merge priques P & Q, )
58     BEGIN ( R is Roving pointer, D is Distance to near leaf )
59         OVER 0= IF      ( Q = nil ? )
60             2 PICK dist@ EXIT THEN ( yes, D = P->dist; done! )
61             2 PICK 0= IF      ( P = nil ? )
62                 ROT DROP OVER SWAP      ( yes, P = Q )
63                 2 PICK dist@ EXIT THEN ( yes, D = P->dist; done! )
64                 2 PICK key @ 2 PICK key @      ( P->key < Q->key ? )
65                 < IF ROT go-rt ROT ROT ELSE      ( yes, P moves right )
66                     SWAP go-rt SWAP THEN      ( no, Q moves right )
67             AGAIN ;      ( loop until one of the trees is eliminated )
68
69 ( This is steps M3: and M4: from Knuth p 619 )
70 : fix-dist ( P Q R D -- P ; fixes up distance to nearest leaf )
71     BEGIN
72         OVER 0= IF 3DROP EXIT THEN      ( R = nil ? yes, done! )
73         ROT DROP OVER rite @ ROT ROT      ( Q = R->rite )
74         OVER left @ dist@ OVER < IF      ( R->left->dist < D ? )
75             DROP DUP left @ dist@ 1+      ( D = R->left->dist + 1 )
76             OVER DUP left @ rite!      ( R->rite = R->left )
77             OVER 4 PICK left! ELSE      ( R->left = P )
78             1+ OVER 4 PICK rite! THEN      ( D++; R->rite = P )
79             DUP 2 PICK dist !      ( R->dist = D )
80             >R ROT DROP SWAP DUP R>      ( P = R; R = Q )
81         AGAIN ;      ( spin down the right sub-tree )
82
83 ( tq-merge is Knuth's Algorithm M from p 619 )
84 : tq-merge 0 list-merge fix-dist ; ( P Q -- P ; merge 2 tq's )
85
86 ( the timer queue root pointer )      VARIABLE TQ 0 TQ !
87
88 ( TQ! updates the father pointer in the first node, & sets TQ )
89 : TQ!      ( tq -- ; sets TQ and father of tq )
90     DUP TQ !      ( set the timer queue head )
91     DUP 0= IF DROP EXIT THEN      ( empty? yes, done! )
92     father TQ SWAP ! ;      ( no, set father of top node )
93
94 : ?waiting DUP father @ @ = ;      ( tq -- flag )
95
96 ( ----- timer queue package entry points ----- )
97
98 : tq-enque      ( &tq -- ; enques to timer )
99     0 OVER left ! 0 OVER rite !      ( nullify both sub-trees )
100    1 OVER dist !      ( distance to a leaf is 1 )
101    TQ @ tq-merge TQ! ;      ( merge new node with old queue )
102
103 : tq-deque      ( -- &tq ; dequeues from timer )
104    TQ @ DUP 0= IF EXIT THEN      ( returns nil on empty queue )
105    DUP      ( save head for answer )
106    DUP left @ SWAP rite @ tq-merge TQ! ;      ( merge remains )
107
108 : tq-unque      &tq -- &tq ; removes from timer )
109    DUP ?waiting NOT IF EXIT THEN 0 OVER father @ ! ( cut )
110    DUP left @ OVER rite @ TQ @ ( both subtrees of tq & TQ )
111    tq-merge tq-merge TQ! ;      ( paste it back together )

```



# Gold Hill delivers GoldWorks.™

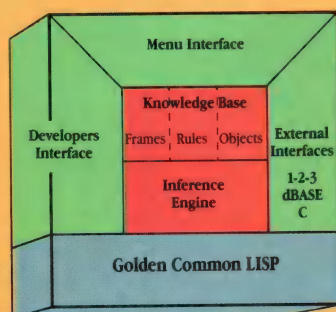




# Now you can build and deliver serious expert systems on advanced PCs.

## GoldWorks changes the economics of expert system building.

GoldWorks, formerly code-named Acorn, is designed for professional software developers who need to build serious expert systems and integrate them with conventional applications running on 286- and 386-based PCs. It combines the best features of high-end expert system tools with the ability to develop and deliver expert systems *inexpensively*, on advanced PCs.



GoldWorks is the most comprehensive expert system tool available for advanced PCs.

## Works like a shell.

GoldWorks gives you the best features of an expert system shell. With the easy-to-use menu interface, you can rapidly prototype and build expert system applications without knowing the underlying programming environment. And you get the GoldWorks tutorial, the San Marco LISP Explorer® tutorial, an on-line help system, and example applications to get you started quickly.

## Works like a toolkit.

GoldWorks gives you the best features of an expert system toolkit. You can access the underlying pro-

gramming environment to extend and customize the system for your specific applications. And you can address up to 15 MB of extended memory on the PC AT (and even more on 386-based PCs).

## Works like expert system tools previously available only on high-end workstations . . . at a fraction of the cost.

GoldWorks sets a new standard for expert system development and delivery on advanced PCs. You get frames with multiple inheritance for flexible knowledge representation. Rules supporting integrated forward and backward chaining for powerful inferencing. Object programming for developing modular applications. Plus advanced features for controlling the inferencing process, including rule sets, sponsors, rule priorities, certainty factors, and extensive rule inspecting and debugging facilities. All on conventional hardware—the PC you already use.

## Works to develop and deliver your expert systems.

GoldWorks is the only tool that lets you develop and deliver serious expert systems on PC ATs. And GoldWorks also takes advantage of PCs based on Intel's powerful 80386 processor, including the COMPAQ DESKPRO 386 and Gold Hill's 386 LISP System.

## Works to integrate expert systems with conventional PC applications.

With GoldWorks, you can integrate expert system applications with dBASE III and Lotus 1-2-3 . . . integrate C routines and libraries into your expert systems . . . and build and deliver expert systems in network environments.

## Works the way you want an expert system builder to work.

GoldWorks from Gold Hill sets the standard by which all other expert system tools will be measured. It was extensively field-tested by developers in dozens of major corporations. And GoldWorks is backed by Gold Hill's comprehensive customer support, training and consulting programs.

Now you have the expert system builder that works the way *you* want to work—GoldWorks. To see how it works, order our unique Demonstration Kit, including full color video and complete User's Guide. It's only \$49 postpaid, refundable with your GoldWorks purchase. To order, call toll-free:

**1-800-242-5477.**

In Mass., call (617) 492-2071.

## GoldWorks from Gold Hill. The expert in AI on PCs.

Gold Hill Computers, Inc.  
163 Harvard Street  
Cambridge, MA 02139



© Copyright 1987 Gold Hill Computers, Inc. Gold Hill, GoldWorks, GCLISP, and 386 LISP System are trademarks of Gold Hill Computers, Inc. San Marco LISP Explorer is a registered trademark of San Marco Associates. Lotus is a registered trademark and 1-2-3 is a trademark of Lotus Development Corporation. dBASE is a trademark of Ashton-Tate. IBM and IBM PC-AT are registered trademarks of International Business Machines Corporation. Intel is a registered trademark and 80286 and 80386 are trademarks of Intel Corporation. COMPAQ and DESKPRO 386 are trademarks of COMPAQ Computer Corporation.



# Two-Bit Analog-to-Digital Conversion

by John Musselman

**T**his article describes a simple technique for measuring an analog voltage that is not only useful but also demonstrates some programming techniques commonly used in real-time applications. The key to such programming is the use of hardware interrupts. A lot of programmers might be wary of hardware interrupts, but once you use them you will find they are a powerful tool. I work with embedded controllers a lot, and so I use them all the time.

## Using Hardware Interrupts

The simplest interrupt scheme utilizes a hardware timer to generate an interrupt at regular intervals. How this is done depends on the hardware in the system, so it is not possible to go into much detail. In general, the idea is to set a timer to its free-running mode. In this mode, the timer is set to some value that decrements on each system clock pulse. When the timer reaches zero, it generates an interrupt, is automatically reloaded, and starts to decrement again. Thus, the interrupt routine is entered at regular intervals. This creates a time base for any routines running in the interrupt and guarantees that each routine in the interrupt will be executed within a certain interval. I almost always have one such interrupt running in any system I design. Many routines that might otherwise qualify for their own interrupt can often

## *An example of putting interrupts to work and how software can directly interact with hardware*

run in this "master" interrupt, which helps simplify things.

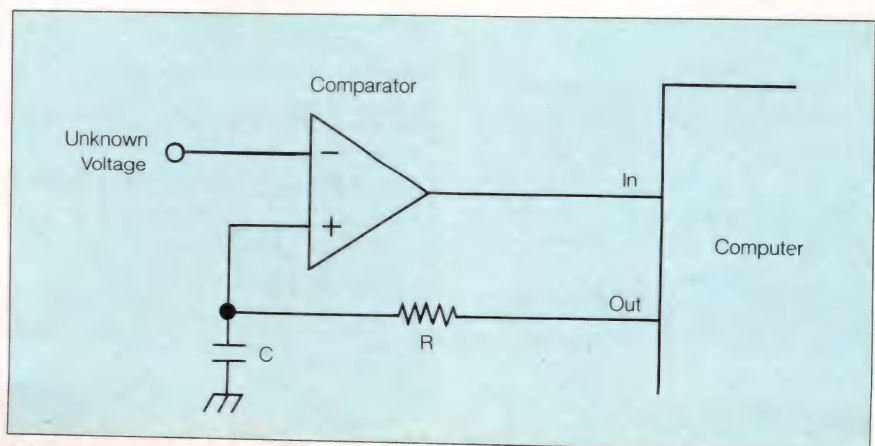
The driver for the analog-to-digital (A-to-D) circuit appears in this kind of interrupt. The frequency of the interrupt is not critical, although the faster it is, the more quickly or more accurately the conversion can be made. Once every millisecond (1,000 times a second) is a convenient figure.

Figure 1, below, is a simplified schematic of the A-to-D converter circuit. Just two bits—one output and one input—interface the circuit to the computer. To understand the circuit, first consider the part of it represented by the output bit, the resistor,

and the capacitor. This is actually a simple D-to-A converter. The output bit is set high or low during each interrupt. The resistor and capacitor values are relatively large, so the voltage on the capacitor is an average of the output voltage over time. More 1s and fewer 0s produce a higher voltage; fewer 1s and more 0s give a lower voltage. If the ratio of 1s to 0s is held constant over time, the voltage will be essentially constant and proportional to the percentage of 1 bits being output.

A simple way, then, to program a D-to-A with one output bit, a resistor, and a capacitor would be to repeatedly output  $m$  0s followed by  $n$  1s, where  $m+n$  is a constant. This could be programmed with a software counter that decrements on every interrupt. Comparing the counter against some variable determines whether to output a high or a low during that particular interrupt. When the counter reaches 0, it is reset to  $m+n$ . The variable thus would control the output voltage as the variable ranges from 0 to  $m+n$ .

The A-to-D converter does not use



**Figure 1:** Simplified schematic of analog-to-digital converter circuit

John Musselman, Route 3, Box 344, Escondido, CA 92025. John is VP of engineering for PMC Industries Inc. and is a consulting product design engineer. He has designed numerous micro-computer-based instruments and controllers.



exactly this method to create its output voltage, but the concept is the same. A counter is used to control the conversion cycle, and the ratio of 1 bits to 0 bits output during each cycle determines the output voltage. The difference is that the output is not all 0 bits followed by all 1 bits; the 1s and 0s can appear in any order.

In Figure 1, notice that the unknown analog voltage is fed into one input of the comparator. The other input of the comparator is the voltage generated by the computer, as explained earlier. The output of the comparator is a digital signal that indicates whether the unknown voltage or the generated voltage is higher.

### The Software Driver

The A-to-D software driver samples the comparator output at each interrupt. If the generated voltage is lower than the unknown voltage, a 1 is output, charging the capacitor to a slightly higher voltage by the next interrupt. If the generated voltage is higher than the unknown voltage, a 0 is output, discharging the capacitor to a slightly lower voltage. In this way, the generated voltage seeks the level of the unknown voltage and then hovers about it. With large resistor and capacitor values, the searching effect is small and the two voltages are essentially equal. You can determine what voltage you are outputting by counting the percentage of 1 bits; this value is the answer sought.

Example 1, right, shows how you do this. The code is for a TMS7000 series processor. It is fairly easy to read, even if you are not familiar with this device, but let me point out a few things. First, the *MOVD* instruction is a 16-bit move. The second byte of the variable is specified as the operand for this instruction, as in *MOVD X+1,Y+1*. Because the processor does not have a 16-bit increment instruction, I have used a 16-bit decrement to count in one's complement. The count is complemented before being saved as the result.

The conversion cycle takes 1,000 1-millisecond interrupts, or 1 second. A 16-bit counter, *COUNT*, keeps track of this time period. Another 16-bit counter, *HIGH*, keeps track of the number of 1 bits output during the cycle. At the end of the cycle, *HIGH* is

```

INBIT: EQU 1 ;INPUT BIT POSITION
OUTBIT: EQU 2 ;OUTPUT BIT POSITION
PERIOD: EQU 1000 ;# OF INTERRUPTS IN A CONVERSION
HIGH: DS 2 ;COUNT OF HIGH BITS
COUNT: DS 2 ;CONVERSION CYCLE COUNTER
RESULT: DS 2 ;RESULT OF CONVERSION

;-----
; INITIALIZATION...
;-----
; MUST BE INCLUDED IN THE SYSTEM
; INITIALIZATION BEFORE INTERRUPTS
; ENABLED
;
MOVD #PERIOD-1,COUNT+1
;
;-----
; INTERRUPT ROUTINES...
;-----
; THE FOLLOWING ROUTINES MUST APPEAR
; IN AN INTERRUPT WHICH OCCURS
; AT REGULAR INTERVALS
;
; I/O...
;
; SEE IF INPUT BIT IS HIGH OR LOW...
;
ATOD: MOVP APORT,B
      BTJO #INBIT,B,HI
      ;
      ; IF IT'S LOW, GENERATED VOLTAGE IS
      ; BELOW THE UNKNOWN VOLTAGE. OUTPUT A
      ; HIGH. COUNT ONE MORE HIGH BIT...
      ;
LO: ORP #OUTBIT,APORT
    DECD HIGH+1 ;NOTE ONES COMPLEMENT COUNT
    JMP IODONE
    ;
    ; IF IT'S HIGH, GENERATED VOLTAGE IS
    ; ABOVE THE UNKNOWN VOLTAGE. OUTPUT A
    ; LOW...
    ;
HI: ANDP #255-OUTBIT,APORT
    ;
IODONE:
    ;
    ;-----
    ; CONVERSION CYCLE...
    ;-----
    ; SEE IF CONVERSION CYCLE DONE...
    ;
    DECD COUNT+1
    JC ATODDN
    ;
    ; IF CONVERSION DONE, SAVE RESULT
    ; AND RESET COUNTERS...
    ;
    MOVD HIGH+1,B ;RESULT IS ONES COMPLEMENT
    COM A ;OF COUNT
    COM B
    MOVD B,RESULT+1
    ;
    MOVD #-1,HIGH+1 ;ONES COMPLEMENT OF ZERO
    MOVD #PERIOD-1,COUNT+1
    ;
    ;
ATODDN:

```

**Example 1:** Analog-to-digital converter driver



# Brand New From Peter Norton A PROGRAMMER'S EDITOR

only  
**\$50**

that's *lightning fast* with the *hot*  
features programmers need

Direct from the  
man who gave you  
*The Norton Utilities*,  
Inside the IBM PC,  
and the *Peter Norton  
Programmer's Guide*.

## THE NORTON EDITOR™



"This is the program-  
mer's editor that I wished  
I'd had when I wrote my  
*Norton Utilities*. You can  
*program your way to  
glory* with *The Norton  
Editor*."

*Peter Norton*



*Easily customized, and saved  
Split-screen editing  
A wonderful condensed/outline display  
Great for assembler, Pascal and C*

Peter Norton Computing, Inc., 2210 Wilshire Boulevard,  
Santa Monica, CA 90403, 213-453-2361. Visa,  
Mastercard and phone orders welcome.

The Norton Editor™ is a trademark of Peter Norton Computing, Inc. © 1986 Peter Norton Computing.

CIRCLE 243 ON READER SERVICE CARD

**The Advanced Programmer's Editor  
That Doesn't Waste Your Time**

# EPSILON

- Fast, EMACS-style commands—completely reconfigurable
- Run other programs without stopping Epsilon—concurrently!
- C Language support—fix errors while your compiler runs
- Powerful extension language
- Multiple windows, files
- Unlimited file size, line length
- 30 day money-back guarantee
- Great on-line help system
- Regular Expression search
- Supports large displays
- Not copy protected

**Only \$195**

**Lugaru**  
Software Ltd.

5740 Darlington Road  
Pittsburgh, PA 15217

**Call  
(412) 421-5911**

for IBM PC/XT/AT's or compatibles

CIRCLE 135 ON READER SERVICE CARD

TWO BIT A TO D  
(continued from page 23)

dumped to *RESULT*, and both counters are reinitialized. *RESULT* can be read by the main routine at any time it is desired to know the input analog voltage. It updates once a second. All this is transparent to the main routine.

The number generated by the A-to-D routine, *RESULT*, can take any value from 0 to 1,000. This means that there are 1,001 possible values—1 more than the number of interrupts in the conversion cycle. The actual voltage that *RESULT* represents is (*RESULT*/1,000) times the voltage swing of the output bit. If you actually build this circuit, be careful to measure this voltage. A TTL output may not provide the full 5 volts that an NMOS or CMOS output will.

Also, there is nothing magic about the number 1,000. The conversion cycle may consist of any number of interrupts. The more interrupts used in a conversion, the greater the resolution but the longer it takes to get the result. You can choose the number of interrupts to suit the application.

This circuit is a good example of how to put interrupts to work and how software can interact with hardware in a direct manner. Using software in a feedback loop, as in this example, is one of the best ways to convert from the analog to the digital world and vice versa.

### Availability

All the source code for articles in this issue is available on a single disk. To order, send \$14.95 to *Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94063 or call (415) 366-3600 ext. 216. Please specify the issue number and format (MS-DOS, Macintosh, Kaypro).

DDJ

Vote for your favorite feature/article.  
Circle Reader Service No. 3.



# SAS Institute Inc. Announces

## Lattice C Compilers for Your IBM Mainframe

### Two years ago...

SAS Institute launched an effort to develop a subset of the SAS® Software System for the IBM Personal Computer. After careful study, we agreed that C was the programming language of choice. And that the Lattice® C compiler offered the quality, speed, and efficiency we needed.

### One year ago...

Development had progressed so well that we expanded our efforts to include the entire SAS System on a PC, written in C. And to insure that the language, syntax, and commands would be identical across all operating systems, we decided that all future versions of the SAS System—regardless of hardware—would be derived from the same source code written in C. That meant that we needed a C compiler for IBM 370 mainframes. And it had to be good, since all our software products would depend on it.

So we approached Lattice, Inc. and asked if we could implement a version of the Lattice C compiler for IBM mainframes. With Lattice, Inc.'s agreement, development began and progressed rapidly.

### Today...

Our efforts are complete—we have a first-rate IBM 370 C compiler. And we are pleased to offer this development tool to you. Now you can write in a single language that is source code compatible with your IBM mainframe and your IBM PC. We have faithfully implemented not only the language, but also the supporting library and environment.

Features of the Lattice C compiler for the 370 include:

- **Generation of reentrant object code.** Reentrancy allows many users to share the same code. Reentrancy is not an easy feature to achieve on the 370, especially if you use non-constant external variables, but we did it.
- **Optimization of the generated code.** We know the 370 instruction set and the various 370 operating environments. We have over 100 staff years of assembler language systems experience on our development team.
- **Generated code executable in both 24-bit and 31-bit addressing modes.** You can run compiled programs above the 16 megabyte line in MVS/XA.
- **Generated code identical for OS and CMS operating systems.** You can move modules between MVS and CMS without even recompiling.
- **Complete libraries.** We have implemented all the library routines described by Kernighan and Ritchie (the informal C standard), and all the library

routines supported by Lattice (except operating system dependent routines), plus extensions for dealing with 370 operating environments directly. Especially significant is our byte-addressable Unix®-style I/O access method.

- **Built-in functions.** Many of the traditional string handling functions are available as built-in functions, generating in-line machine code rather than function calls. Your call to move a string can result in just one MVC instruction rather than a function call and a loop.

In addition to mainframe software development, you can also use our new cross-compiler to develop PC software on your IBM mainframe. With our cross-compiler, you can compile Lattice C programs on your mainframe and generate object code ready to download to your PC.

With the cross-compiler, we also offer PLINK86™ and PLIB86™ by Phoenix Software Associates Ltd. The Phoenix link-editor and library management facility can bind several compiled programs on the mainframe and download immediately executable modules to your PC.

### Tomorrow...

We believe that the C language offers the SAS System the path to true portability and maintainability. And we believe that other companies will make similar strategic decisions about C. Already, C is taught in most college computer science curriculums, and is replacing older languages in many. And almost every computer introduced to the market now has a C compiler.

### C, the language of choice...

C supports structured programming with superior control features for conditionals, iteration, and case selection. C is good for data structures, with its elegant implementation of structures and pointers. C is conducive to portable coding. It is simple to adjust for the size differences of data elements on different machines.

### Continuous support...

At SAS Institute, we support all our products. You license them annually; we support them continuously. You get updates at no additional charge. We have a continuing commitment to make our compiler better and better. We have the ultimate incentive—all our software products depend on it.

### For more information...

Complete and mail the coupon today. Because we've got the development tool for your tomorrow.



SAS Institute Inc.  
SAS Circle, Box 8000  
Cary, NC 27511-8000  
Telephone (919) 467-8000 x 7000

#### I want to learn more about:

- ☐ the C compiler for MVS software developers
- ☐ the C compiler for CMS software developers
- ☐ the cross-compiler with PLINK86 and PLIB86

#### today...so I'll be ready for tomorrow.

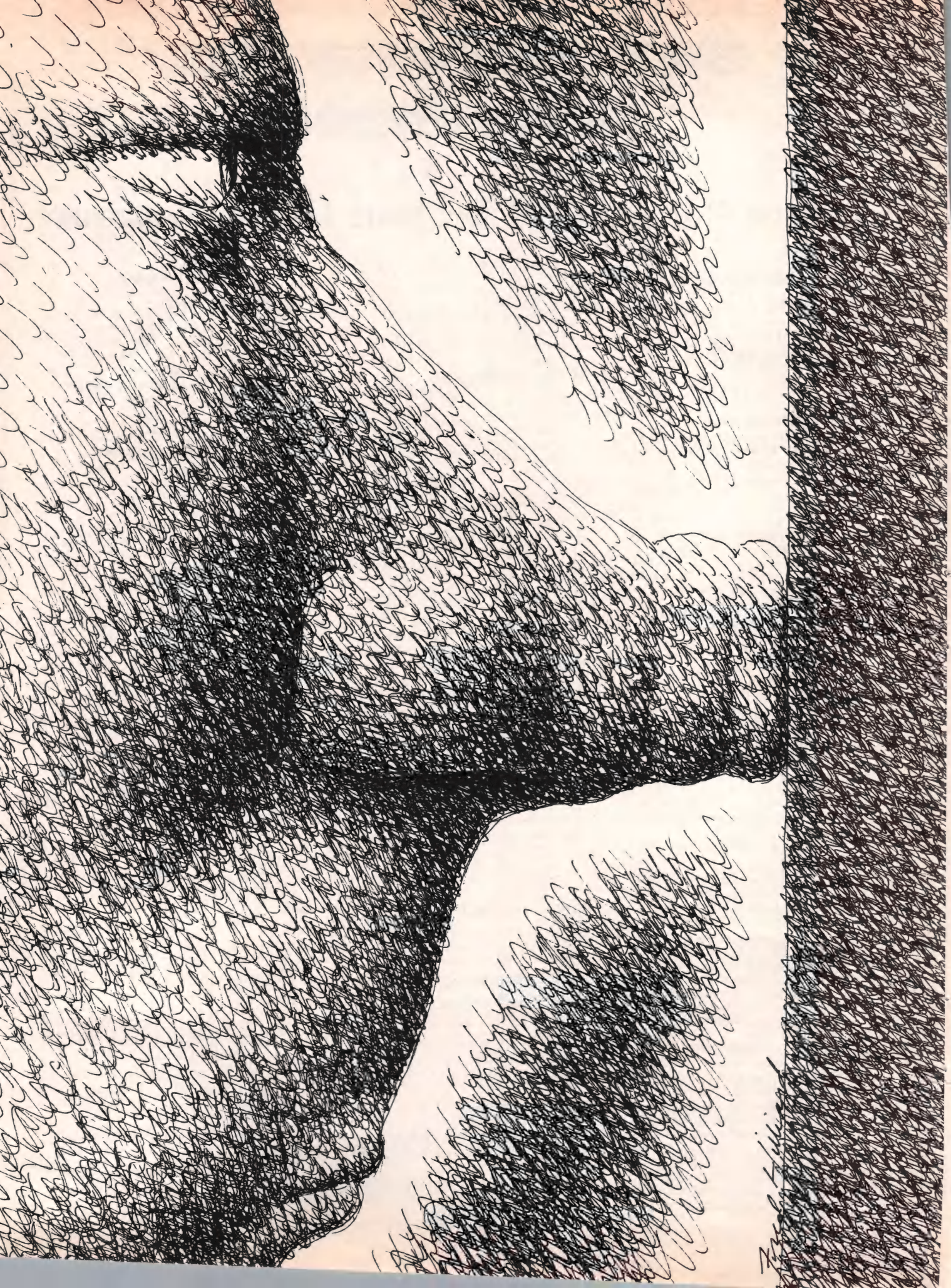
Please complete or attach your business card.

Name \_\_\_\_\_  
Title \_\_\_\_\_  
Company \_\_\_\_\_  
Address \_\_\_\_\_  
City \_\_\_\_\_ State \_\_\_\_\_ ZIP \_\_\_\_\_  
Telephone \_\_\_\_\_

Mail to: SAS Institute Inc., Attn: CC, SAS Circle, Box 8000, Cary, NC, USA.  
27511-8000. Telephone (919) 467-8000, x 7000

DDJ 6 / 87







# Does your programming language present you with annoying "dead ends?"

## It may be time to discover the power of Revelation.

When sophisticated programmers first tried database programming languages they said "Too wimpy!" Now they're discovering Revelation's R/BASIC. Yes! It's a Revelation!

## Let's start by examining R/BASIC philosophically.

First, it's a structured language, interfacing smoothly with DOS as well as C and Assembler. You'll find commands for structural coding like CASE, LOOP/REPEAT with WHILE/UNTIL, IF/THEN/ELSE and other familiar friends here. And, like all serious programming languages, it has a compiler for fast program execution. A compiler so fast that it can translate 20 lines per second for speedy error detection and correction.

## You should expect excellent string handling in a database programming language, but math is another story. Or is it?

Because R/BASIC was developed as a database programming language, naturally you expect it to have powerful string handling capabilities. It does. Revelation's R/BASIC supports both dimensioned arrays and dynamic arrays. Individual strings can be up to 64K in length. But, what you might not expect in a database programming language is a high level of math support. Revelation has it.

R/BASIC offers integral 8087 support for maximum precision. It automatically accesses the 8087 chip for calculations. And, if you don't have an 8087 chip, R/BASIC

incorporates an excellent 8087 emulator. Trig functions? Of course.

## Subroutine flexibility.

Revelation lets you declare your own internal or external subroutines with argument passing. Subroutines can be developed in C and Assembler, then called directly from an application. And with Revelation's recursive routine calling, you won't have to write as much code.

How much less code?

A long COBOL program typically runs 20,000 lines. A long R/BASIC program typically runs 300 lines. Imagine how much of your development time that kind of difference can save.

R/BASIC will also help you produce a clean program fast. Its completely interactive symbolic debugger traces problems in minutes instead of hours.

## Here's another way you can save time with R/BASIC.

By incorporating LOCK and UNLOCK statements in your programs you can change from single to multi-user systems without making program modifications or recompiling.

## Is it time for your personal Revelation?

Once you get used to something, it's difficult to change. But if change means progress, it's worth the effort. R/BASIC is a powerful programming language that offers significant built-in database advantages over a "pure" programming language. These advantages can save time during program development and time when the

application runs. For the serious programmer, time is money. It's worth your time to take a look at Revelation.

## The basics of R/BASIC.

- Compiler.
- Internal and external subroutine calling with argument passing.
- Recursive routine calling.
- Integral 8087 support for maximum precision.
- Support for structured programming.
- No data typing.
- English variable names.
- User-defined functions.
- Full screen editor.
- Support for ASM and C routines.
- Sophisticated string handling includes dynamic array support with a 64K limit.
- Trig functions.
- Alternate syntaxes.
- Direct DOS interface.
- Completely interactive debugger.

If you would like to sample the power of Revelation, please send \$24.95 for our comprehensive Demo/Tutorial. A phone call gets you complete information.

# COSMOS™

Cosmos, Inc.  
3633 136th Place S.E.  
Bellevue, WA 98006, USA  
Phone (206) 643-9898  
Telex: 185210 (COSMOS MUT)  
FAX: (206) 643-7609



# The XOR Chain

by David E. Cortesi

In a system in which unsigned binary words and storage addresses are compatible types, a peculiarity of the exclusive-OR function allows both links of a doubly linked list item to be stored in a single word. This space-saving gimmick has been part of the folklore of computing for some time. In this article I explore the mathematical basis of the trick and develop a package of C functions to implement it in a modular style.

## The Exclusive-OR Function

Consider the exclusive-OR function (XOR). You can think of XOR as an arithmetic function similar to addition and subtraction; it obeys similar metarules. Like them, for instance, it possesses 0 as an identity element. The identity element of a function is the value whose use preserves an identity over the function. For addition:

$$A + 0 = A$$

That is, adding 0 is an identity operation; it leaves A unchanged. In subtraction:

$$A - 0 = A$$

$$A - A = 0$$

Here, not only does subtraction of 0 leave an identity unchanged, but subtracting identical values yields the identity element. Just so:

$$A \text{ XOR } 0 = A \quad (1)$$

$$A \text{ XOR } A = 0 \quad (2)$$

Relation (2) is the basis of the common

**You can  
construct a  
doubly linked list  
with both links in the  
same word.**

assembly-language trick of clearing a register by XORing it with itself.

The addition operation is commutative—that is:

$$A + B = B + A$$

This is not true of subtraction, but it is true of XOR:

$$A \text{ XOR } B = B \text{ XOR } A \quad (3)$$

Addition is also associative—that is:

$$(A + B) + C = A + (B + C)$$

Again, this is not true of subtraction but is true of XOR:

$$A \text{ XOR } (B \text{ XOR } C) = (A \text{ XOR } B) \text{ XOR } C \quad (4)$$

The XOR operation differs from addition and subtraction in that all the relations (1) to (4) hold for it, whereas only (1) and (2) are true of subtraction and only (1), (3), and (4) hold for addition. Because all four relations hold for XOR, so does this peculiar rule:

$$(A \text{ XOR } B \text{ XOR } C) \text{ XOR } B \text{ XOR } C = A \quad (5)$$

This may not be immediately obvious. Come at it in steps: apply relations (3) and (4) several times to rearrange the formula so it reads:

$$A \text{ XOR } (B \text{ XOR } B) \text{ XOR } (C \text{ XOR } C) = A$$

By relation (2), this is equivalent to:

$$A \text{ XOR } (0) \text{ XOR } (0) = A$$

but by (1), 0 XOR 0 is just 0, leaving you with:

$$A \text{ XOR } 0 = A$$

which merely restates (1).

## Three Links in a Word

Mathematically, (5) is a tautology, a gassy expansion of (1), but it has a practical use for data storage. From it follows this: if you have a binary word W that contains the result of:

$$W = (A \text{ XOR } B \text{ XOR } C)$$

and if you know any two of these values independently, you can recover the third value from W. By relation (5):

$$W \text{ XOR } B \text{ XOR } C = A$$

Similarly, you can recover B knowing A and C, or C knowing A and B. You can use this fact to construct a doubly linked list using but a single binary word to store both links in each list item.

How do you do this? In a doubly linked list, each list item contains the addresses of its predecessor and its successor. Call these A and C, and use the item's own address as B. Set its single link word W to:

$$W = (A \text{ XOR } B \text{ XOR } C)$$

Now, if a program is examining a list item, it obviously must know that item's address. Provided it arrived at

David E. Cortesi, 415 Cambridge St., #18, Palo Alto, CA 94306. Dave is a former DDJ columnist.



the item by tracing the list, it must have come down to it from its predecessor or back from its successor, and hence it must know one of the addresses A or C. Therefore, by (5), it can recover the unknown address from word W and use it to proceed onward in the list.

That's the basic idea. A complete implementation, however, must handle the cases of the empty list and lists of one and two items, should allow for inserting and deleting elements, and ought to be packaged as functional subroutines for simplicity.

Here I'll develop such a package in C. In the following code, I assume that the unsigned binary integer is exactly compatible with a C pointer (as may not be the case in, for instance, the Intel 8086 with a large-memory model), and I've left out the casts that the more picky C compilers may want inserted to make that equivalence manifest. The code favors clarity over absolute efficiency, so you may wish to insert register declarations and other changes for speed.

### Defining an Item

In C, the XOR function is denoted by an up arrow and inversion of a Boolean value by comparison to 0. For readability, assume that the following *defines* hold in the example code:

```
#define Xor ↑
#define Not 0==
#define Nil 0
```

Now you can define the structure of a list item:

```
struct Item {
    unsigned link;
    /* other contents of item */
};
```

The link word contains the XORed linking addresses. What the "other contents" might be depends on the application, and so I assume that functions to create and destroy *Items* are defined elsewhere:

```
extern struct
    Item *MakeItem();
extern void
    DropItem(i) struct Item *i;
```

### Defining a List

A list consists of a chain of zero or more *Items*, but there has to be an anchor for the chain: a single fixed place where the head and tail of the list can be found. I define its form as *Anchor*:

```
struct Anchor {
    struct Item *head, *tail;
};
```

For any list there will be just one *Anchor*. Furthermore, I legislate that if a list is empty:

**You can  
think of XOR  
as an arithmetic  
function.**

```
(head == Nil) && (tail == Nil)
```

Otherwise, *head* points to the first item of the list and *tail* points to the last item. If the list contains but one item:

```
(head == tail) && (head != Nil)
```

You can express these rules in Boolean functions that test an *Anchor*:

```
int ZeroItems(a) struct Anchor *a;
{ return(
    (Nil == a->head) &&
    (Nil == a->tail)
); }
```

```
int OneItem(a) struct Anchor *a;
{ return(
    (a->head == a->tail)
    && (a->head != Nil)
); }
```

### Scanning a List

The *Anchor* of a list gives you access only to the first and last items. If you are to reach intermediate items, you must scan over the list, either forward from the head or backward from the tail.

While scanning the list, your position is always maintained in a pair of pointers that contain the addresses of two items that are logically adjacent in the list. The item that is nearer the head of the list I call the *prior* item; the one closer to the tail I call the *next* item. When I have these items set up with valid addresses, I refer to such a pair of pointers as a *Scan*, because I can use the pair to scan over the list in either direction. Because a pair of pointers is always required, I put them in a record. Because I am always scanning a particular list, I include the address of the list's *Anchor*:

```
struct Scan {
    struct Item *prior, *next;
    struct Anchor *base;
};
```

Although a list may have only one *Anchor*, it may have any number of *Scans* associated with it. No *Scan* is valid until it has been associated with some list, however. Here's a procedure to do that:

```
void Associate(s,a)
struct Scan *s;
struct Anchor *a;
{ s->base = a; }
```

This procedure is very simple, but in some applications, it might have to do more. It might be desirable, for example, to keep track of the number of scans that are active on a list, and the *Associate* function could do that by incrementing a count in *Anchor*.

A program may scan a list by moving a *Scan* structure from the current position to an adjacent one. But a scan has to start somewhere. This procedure sets a scan to the head of the list:

```
void ToHead(s) struct Scan *s;
{
    s->next = s->a->head;
    s->prior = Nil;
}
```

When a scan is at the head, the *prior* pointer contains *Nil* (nothing precedes the head of a list) and the *next* item is the head item of the list. A simple function can test a *Scan* for this condition:

```
int AtHead(s) struct Scan *s;
{ return( s->prior == Nil ); }
```



## XOR CHAIN

(continued from page 29)

A procedure can set a scan to the tail of its list:

```
void ToTail(s) struct Scan *s;
{
    s->prior = s->a->tail;
    s->next = Nil;
}
```

When a scan is at the tail of a list, the *next* pointer contains *Nil* (nothing follows the tail of a list) and the *prior* item is the tail item of the list. A Bool-

ean function can test for this condition:

```
int AtTail(s) struct Scan *s;
{ return( s->next == Nil ); }
```

Recall that a list is empty when the head and tail pointers in its *Anchor* are *Nil*. Verify for yourself that, for a *Scan* associated with an empty list, applying either *ToHead* or *ToTail* makes both *AtHead* and *AtTail* report true. Earlier I defined a function *Zeroltems*, which tests an *Anchor* to see if it's empty. Now I can define a function that tests not an *Anchor* but a

*Scan*:

```
int EmptyList(s) struct Scan *s;
{ return( AtHead(s) &&
    AtTail(s)); }
```

Consider the implications for loop control. Either the sequence:

```
ToHead(z);
while (Not AtTail(z))
    {step forward in list}
```

or the sequence:

```
ToTail(z)
while (Not AtHead(z))
    {step backward in list}
```

can work safely—that is, do nothing—on an empty list.

### Stepping a Scan

Having set a *Scan* to the head of a list, you want to be able to step it forward in the list. You do that using the identities of XOR. Consider an *Item* that is neither at the head nor the tail of a list. If it is located at address B and if its predecessor is at address A and its successor at address C, then its link word contains A XOR B XOR C. The link word plus the contents of the *Scan* let you step it forward:

```
void GoFwd(s) struct Scan *s;
{ struct Item *i;
    i = s->prior Xor
      s->next Xor
      s->next->link;
    s->prior = s->next;
    s->next = i;
}
```

Variable *i* receives the link word of the next *Item* minus the contributions made by its own and its predecessor's address: in short, its successor's address.

In this way you can process all the items of a list:

```
Associate(z,a);
ToHead(z);
while( Not AtTail(z) )
{
    Process(z->next);
    GoFwd(z);
}
```

If you are accustomed to C, you may prefer to see this written using a *for*

## Never Miss A Compile Again!

BSW-Make, our retargetable *make* utility, speeds software development by automating the chore of rebuilding complex software products after an editing session. No more missed compiles! No more wholesale "just in case" recompilations of the whole product! BSW-Make insures that the minimum set of compilations, assemblies, and links required to correctly update your software are performed after each edit. A major timesaver!

- Syntax compatible with UNIX *make*
- Works with any compiler, assembler, or linker
- Macro facility for parameterized builds
- Indirect command file generation facility overcomes operating system command length limitations
- MS-DOS/PC-DOS version only \$89.95
- VAX/VMS version from \$299.95
- Not copy protected
- Unconditional 30-day guarantee — try it at no risk!

for free product information, call

**(617) 367-6846**

Ask for Department D2

**The Boston Software Works, Inc.**

120 Fulton Street, Boston, MA 02109

CIRCLE 384 ON READER SERVICE CARD



# Develop DB applications 10 times faster without the coding pain...you'll swear it's Magic

AKER Corp. **MAGIC PC** 12/03/86

13. Order Entry Screen

Execution Definition

Change	Description	Prefix	Main	Suffix
1	Record	--	42	8
2	Task	--	--	1

Op	Operation	Type	No.	Description	Assign	Exp	Key
30	3	Req. Link	>	File	2	Customers	0
31	1	Sel. Field	>	R	2	Customer Name	0
32	1	Sel. Field	>	R	4	Customer Discount	0
33	4	End Link	>				
34	0						
35	8	Exec. Prog	>	No	18	Item List	2
36	0						
37	9	Upd. Field	>	No	8	Customer Discount	3
38	0						
39	7	Exec. Task	>	No	1	Order Lines	0

1>Opt 2>Undo 3>Del 4>Add 5>Zoom 6>Expr 7>Draw 8>Task 9>End 10>Help

Visual Programming with Magic PC looks as simple as this. To design an application you quickly place your design specifications into menu-driven Task Tables without having to write a single line of code. For example, just by highlighting the Execute Program operation on the left screen and also highlighting the Item List

Order Entry

Order No: 999 Order Date: 99/99/99 Customer No: 99999 Address: AAAAAAAAAAAAAAAAAA

Line	Item	Type	Description	Quantity	Unit Price	Total Price
999	99999	A	AAAAAAAAAAAAAAAAAAAA	9.999	999.999 99	999.999 99

Item List

No.	Description	Type	Price
999	AAAAAAAAAAAAAAAAAAAA	A	999.999

Stock Status

In Stock: -999.999  
Total Orders: -999.999  
Avail to Sell: -999.999

	Order Sum	Sub-Total	Sales Tax	Order Total
99.999	-999.999 99	-999.999 99	-999.999 99	-999.999 99

1>Opt 2>Undo 3>Del 4>Add 5>Zoom 6>Expr 7>Draw 8>Task 9>End 10>Help

program in the Program Menu, you tell Magic PC to Zoom into the Item List program through the window shown on the right screen. The window will automatically scroll the Item List data horizontally and vertically, and allow query, "cut and paste" copy or even creation of new Items.

## Free yourself from coding

Database professionals throughout the world are discovering a new way to dramatically cut development time.

So can you! With Magic PC, the Visual Database Language by Aker.

Consultants, VAR's, Software Houses and DP MIS professionals: If you develop DB applications for a living, now you can tackle any database application 10 times faster than with your DBMS or 4GL.

What makes you so fast with Magic PC? It's not magic...it's simply because Magic PC finally frees you from coding. And doesn't coding take up most of your time right now?

Magic PC lets you leverage your design skills instead of wasting your time coding. Now you can generate a fully functional prototype in just hours for quick customer feedback, and easily refine the same prototype to a finished application.

All you do is enter your system design specifications directly into Magic PC's non-procedural menu-driven Tables, as ideas come to mind, and Magic PC generates the programs for you automatically.

Magic PC gives you a free hand to design powerful data management systems limited only by your own imagination. Without the time consuming mechanical details of conventional procedural programming. There's your competitive edge. The rest is up to you.

Your biggest time saving comes from Magic PC's dynamic adaptation to spontaneous design changes. You're free to change your design on the fly, and Magic PC automatically updates your programs and data files online. No more time wasted maintaining each program manually with every small change.

## Visual Programming Power

You program with Magic PC by describing your data elements with Data Dictionary Tables (Files, Fields, Keys), and placing your system design specifications into Task Description Tables.

The Tasks can be nested within one another or dynamically Link to satellite Tasks, to give you true One-to-Many relational database power.

Only 13 Task Operations harness the power of Magic PC. Operations are specific enough to eliminate the need for tiresome coding, yet elastic enough to produce robust custom applications.

Use the Task building blocks to quickly generate Online Programs: Screens, Window Zooms, Menus; or Batch Programs: Reports, Updates, Data Import/Export and much more.

You develop the Task Tables visually on the screen by highlighting selections from Window Zooms and pop-up menu-driven Tables. You're not forced to follow any particular Table sequence, and there's no coding to slow you down. It's that simple.

You can apply mathematical and logical Expressions, or use the built-in Functions directly in the Task Tables to automate conditional Task processing, to display custom error messages or even invoke external applications such as spreadsheet, word processing

or communication programs, transparently from within your Magic PC application programs.

Magic PC generates your application by fusing all your Data Dictionary and Program Tables seamlessly into a single Integrated Library, and automatically maintains changes online for optimal, bug-free performance, so you always get it right the first time.

Your application is executed at runtime by a Magic Run engine for stand-alone operation, and you can distribute your applications at a low cost and protect your design. Magic Run has a built-in visual interface to manipulate data and get on-the-spot ad-hoc information without any commands or syntax, simply by highlighting selections from menus. Data validation, security and error-checking are done automatically for you without programming.

Magic PC has built-in support for File and Record Locking so you can design multi-user applications for a local area network, and share data with any number of Magic Run users.

Magic PC integrates the Btrieve file manager internally, supporting the B-Tree file structure for fast high performance data access, and fault tolerant recovery during power failures.

Magic PC's powerful Window Zoom lets you design composite screens with windows to probe deep into the application through nested windows and manipulate the data underneath. By Zooming from any field, your end-user can conveniently query, copy or even create data in other programs directly through the windows, without stopping their screen session. The window frame size does not limit the available Data View since each window has built-in horizontal and vertical scrolling.

## Magic PC is the professional's choice:

IBM France: "IBM encourages Magic PC and salutes such evolution..."

Israeli Air Force: "We were convinced that it was not possible to have a design tool powerful enough to implement real life applications without a programming language. Magic PC changed our mind..."

PC Magazine: "If the thought of programming database applications makes you tremble, Magic PC is for you. The applications generator saves users from the need to deal with much dreaded computer language code..."

PC Tech Journal: "Magic PC is probably the best integrated database application and screen generator that we have seen...very smooth system, and smoothness comes at a premium these days..."

PC World: "Relational data managers and application generators that offer power without programming are a bit like perpetual motion machines - very rare. Into that vacuum comes Magic PC, a data management tool without language that is ideal for turnkey applications..."

PC Week: "Rather than use a written programming language the user is given a great deal of freedom and power to create complex relational database applications...this package is a true time-saver..."

## Get your Magic Tutorial for only \$19.95

See for yourself how fast and powerful visual programming can be for you. Order your copy of Magic PC Tutorial including Tutorial disks, and a step-by-step tutorial to quickly set up an Order Entry application without coding. All for only \$19.95 and We'll credit the Tutorial cost towards the \$695 Magic PC price for up to 30 days.

## Or save \$500 at no risk!

Yes, for a short time only, qualified VAR's can save almost \$500 off the list price and get the complete unprotected Magic PC software and documentation at a special VAR price of \$199\* with no risk. Keep it at this low price only if it's as good as we say, or return it within 30 days for a full refund if you're not completely satisfied. Act now and you'll also get 2 Magic Run (runtime modules) for only \$95.

## Order now for immediate delivery

Call this toll-free number now with your credit card or COD charge, or send the Order Coupon below today with your check to Aker.

**1-800-345-MAGIC**  
in CA 714-250-1718



Yes, please rush me the following  
Prices include 2nd day shipping

- |  |                 |
|--|-----------------|
| <input type="checkbox"/> Magic PC Tutorial     | \$ 25.95        |
| <input type="checkbox"/> Magic PC (VAR's only) | \$209.00        |
| <input type="checkbox"/> Magic PC              | \$705.00        |
| <input type="checkbox"/> Magic Run             | \$100.00        |
| in CA add 6% tax                               | \$ _____        |
| <b>Total</b>                                   | <b>\$ _____</b> |

Ship to: \_\_\_\_\_  
Address: \_\_\_\_\_  
City/ST/Zip: \_\_\_\_\_  
Phone: \_\_\_\_\_

OEM inquiries are welcome. Prices valid in North America only.  
\*Less \$19.95 restocking fee. Limit one per customer, subject to availability. Not for resale.  
System requirements: IBM PC, XT, AT and 100% compatibles, PC-DOS 2.0 or later, 512K and hard disk.

**AKER**

Aker Corp. 18007 Skypark Cir. B2, Irvine, CA 92714  
Tlx 4931184 AKER UI  
Aker S.A. 11 Route de Florissant CH-1206, Geneva  
Switzerland Tlx 421792 AKER CH

Trademarks: Magic PC, The Visual Database Language, Window Zoom, Magic Run, Magic LAN and Magic PC Tutorial are trademarks of Aker Corp., IBM PC and PC-DOS are trademarks of IBM Corp., Novell is a trademark of Novell Inc., Btrieve is a trademark of Softcraft Inc.



## WINDOWS FOR DATA™

# The first choice of professional C programmers

"Windows for Data is the best  
programming tool I've ever used.  
It's the most flexible I've seen.  
Whenever I've wanted to do something,  
I've been able to find a way."

Steven Weiss,  
Stratford Systems

Professionals choose our tools because they are designed, crafted, and supported for professionals. Here at Vermont Creative Software, we understand that performance and pleasure in programming derive from more than a long list of functions. **Windows for Data** provides:

### PROFESSIONAL FLEXIBILITY:

Our customers repeatedly tell us how they've used WFD in ways we never imagined - but which we anticipated by designing WFD for unprecedented adaptability. Virtually every capability and feature can be modified to meet special needs. You will be amazed at what you can do with WFD.

### PROFESSIONAL PERFORMANCE:

Screen output is crisp and fast. Windows, menus, and data-entry forms snap up and down from the screen. WFD is built upon and includes **Windows for C**, the windowing system rated #1 in speed and overall quality in PC Tech Journal (William Hunt, July 1985).

**PROFESSIONAL RELIABILITY:** An unreliable tool is worse than no tool at all. VCS products are known in the industry for their exceptional reliability. Ask anyone who owns one.

**PROFESSIONAL DOCUMENTATION:** Over 600 pages of documentation provide step-by-step explanations for each major application, a reference page for each function, listings of functions alphabetical-ly and by usage, and a fully cross-referenced

index. Extensive tutorials and demonstration programs assist learning.

**PROFESSIONAL TECHNICAL SUPPORT:** The same expert programmers that develop our products provide prompt, knowledgeable technical support.

**PROFESSIONAL PORTABILITY:** High-performance versions of VCS products are available for XENIX, UNIX, and VMS, as well as DOS. No royalties on end-user applications.

### OUR CHALLENGE AND GUARANTEE

If you have an application where no other tool can do the job, try **Windows for Data**. If it doesn't help you solve your problem, RETURN FOR A FULL REFUND. YOU MUST BE SATISFIED.

Ask for **FREE DEMO DISKETTE**



**Vermont  
Creative  
Software**

21 Elm Ave.  
Richford, VT 05476  
Telex: 510-601-4160 VCSOFT  
**Tel.: 802-848-7731**

Prices: PC DOS\* \$395; XENIX, VMS, UNIX  
\*PC DOS specify C compiler.

## WINDOWS FOR DATA

for DOS, UNIX, VMS ...

The complete windowing data entry, menu, and help system that does the hard job others can't - we **guarantee** it!

Pop-up data entry windows; field types for all C data types, plus decimals, dates, and times; auto conversion to and from strings for all field types; system and user supplied validation functions; range checking; required, must-fill, and protected fields; free-form movement; multiple-choice field entry; scrollable sub-forms. Branch and nest windows, forms, and menus.

Complete context sensitive help system with pop-up windows and scrollable text.

Pop-up, pull-down, scrollable, and Lotus-style menus.

**NEW FOR DEBUGGING:** Exclusive **VCS Error Traceback System** automatically identifies the location and cause of program errors. Eliminates the need to code error checks on all function calls! **VCS Memory Integrity Checking** helps catch those hard-to-detect, memory-corruption errors.

**NEW FOR ERROR HANDLING:** Install your own error handler to be called whenever a function detects an error.

**NEW FORM LAYOUT UTILITY** simplifies form design.



# THE ADA<sup>®</sup> WORLD JUST CHANGED.

Meridian Software Systems, Inc. announces the world's first pre-validated Ada compiler for PC-class machines requiring no additional hardware. The Meridian AdaVantage™ compiler hosted on IBM PC/XT/AT and compatibles running MS-DOS was pre-validated using ACVO version 1.8.

The introductory price is \$395.00. After July 1, 1987, the list price will be \$795.00.



23141 Verdugo Drive, Suite 105  
Laguna Hills, California 92653  
714/380-9800

Ada is a registered trademark of the U.S. Government (A.J.P.O.). AdaVantage is a trademark of Meridian Software Systems, Inc. References to other computer systems use trademarks owned by the respective manufacturers.

CIRCLE 397 ON READER SERVICE CARD



# XOR CHAIN

(continued from page 30)

loop:

```
Associate(z,a);
for(ToHead(z);
    Not AtTail(z);
    GoFwd(z))
    Process(z->next);
```

which amounts to the same thing.

When you process a list from head to tail, the *Items* to be processed are those that turn up in the next posi-

tion. When you go backward, from tail to head, you take them from the prior slot. Here's a going-backward procedure:

```
void GoBak(s) struct Scan *s;
{ struct Item *i;
  i = s->prior Xor
    s->next Xor
    s->prior->link;
  s->next = s->prior;
  s->prior = i;
}
```

Here you take the link word of the

predecessor *Item* minus the contributions of its own address and of its successor: in short, the address of its predecessor. With this procedure and *ToTail*, you can process a whole list:

```
Associate(z,a);
ToTail(z);
while( Not AtHead(z) )
{
  Process(z->prior);
  GoBak(z);
}
```

It may not be clear that these procedures work for all items of all lists. Before you can be sure they do, you need to define what the link words of the head and tail items contain. The simplest rule works: the link word contains *xOR 0* for irrelevant addresses. Table 1, below, shows a complete four-item list whose anchor is *Test*. Presume that the following loop is to be executed using the list in this table:

```
Associate(z,Test);
ToHead(z);
while(Not AtTail(z)) GoFwd(z);
```

Trace the contents of *prior* and *next* at each step and assure yourself that all the pointers work out correctly. (Surely you don't take articles such as this one on faith, without at least desk-checking the code?) Then trace them through this loop to verify the rest of the code given so far:

```
for(ToTail(z,Test);
    Not AtHead(z);
    GoBak(z));
```

What would happen if a program applied *GoFwd* just once too often? Or *GoBak*? Clearly you ought to account for these end effects. What shall be done with the head item (which has no predecessor) and the tail item (no successor)? You can either wrap around or stick. Here is a safe *StepFwd* procedure:

Address	Contents
Test.head	A
A->link	A xor B
B->link	A xor B xor C
C->link	B xor C xor D
D->link	C xor D
Test.tail	D

**Table 1:** A four-item list whose anchor is *Test*

# Australia's Finest C Compiler

main (argc, argv) 00110101100

**\$129**

plus shipping

## HI TECH C Compiler

- Complete production quality compiler
- Smallest, fastest code from any compiler
- High performance C Compiler for the Z80, 68000, 65816, and 8086 processors
- Runs on CP/M-80, PC-DOS, MS-DOS, CP/M-86, CONCURRENT CP/M, ATARI ST and APPLE II gs
- Now in use at thousands of sites worldwide, including Australian Government and large institutions.
- Excellent user interface
- ROM code is supported and it includes a macro assembler, linker, librarian, object code converter, cross reference utility and full library source code. The 8086 compiler supports large and small memory models and the 8087

**\$199**

plus shipping

## Cross Compilers

- Run under MS-DOS, UNIX, and CP/M-86 and produce code for the 68000, 8086/286, 65816, 8096 and Z80 processors. Each compiler includes an assembler, linker, librarian, object code converter and cross reference utility.



The Cutting Edge

Order from: **SOFTFOCUS** 1343 Stanbury Drive,  
Oakville ONTARIO Canada L6L2J5  
(416) 825 0903 or (416) 844 2610

U.K. Greymatter (0364) 53 499

Australia HI TECH SOFTWARE  
P.O. Box 103 Alderley 4051 (07) 38 6971



CIRCLE 376 ON READER SERVICE CARD



```
void StepFwd(s) struct Scan *s;
{ if (Not AtTail(s)) GoFwd(s); }
```

that simply sticks when it reaches the tail. Here is a *StepBak* procedure that wraps around:

```
void StepBak(s) struct Scan *s;
{
    if (AtHead(s)) ToTail(s);
    else GoBak(s);
}
```

### Inserting an Item

You construct a list by starting with *Zeroltems* true and inserting new items. Example 1, below, shows one way to insert a new item between the prior and next items of a scan. In the statement:

```
s->prior->link Xor=
    (i Xor s->next)
```

```
void Insert(i,s)
struct Item *i;
struct Scan *s;
{
    if (AtHead(s))
        s->a->head = i;
    else
        s->prior->link Xor=
            (i Xor s->next);

    if (AtTail(s))
        s->a->tail = i;
    else
        s->next->link Xor=
            (i Xor s->prior);

    i->link =
        s->prior Xor s->next Xor i;
    s->prior = i;
}
```

**Example 1:** A way to insert a new item between the prior and next items of a scan.

Address	Contents
Test.head	Nil
Test.tail	Nil
A.link	?
B.link	?
C.link	?
D.link	?

**Table 2** An empty list and four prepared items

you take the predecessor's link, remove from it the contribution of the successor's address (by XORing with *s->next*); and install the value of its new successor—the item with address *i*.

A similar statement fixes the link word in the next item, except that it's the predecessor's address that is removed in favor of *i*. Finally, the new item's link is formed from its own address and that of its new predecessor and successor. It becomes the prior item, so a series of insertions will build up in head-to-tail

order.

To verify this, commence with an empty list and four prepared items, as shown in Table 2, below, and execute the following sequence of operations by hand:

```
Associate(z,Test);
Insert(A,z);
Insert(D,z);
GoBak(s);
Insert(B,z);
Insert(C,z);
```

keeping track of the new contents of









## OASYS Solves the Cross-Development Puzzle

*Every Piece is in Place*

# 68020+ 68881

**Oasys offers the complete development solution  
Fast, Highly Optimized and Available**

### 68020 + 68881 and 68000/10 Cross & Native Development Tools

- |  |  |
|--|--|
|  <b>COMPILERS</b><br>• C • C++ • Pascal • FORTRAN |  <b>PERFORMANCE ANALYSIS TOOLS</b>            |
|  <b>MACRO ASSEMBLER/LINKERS</b>                   |  <b>REAL-TIME OPERATING SYSTEMS</b>           |
|  <b>SIMULATORS</b>                                |  <b>LANGUAGE-SENSITIVE EDITORS</b>            |
|  <b>SYMBOLIC DEBUGGERS</b>                        |  <b>COMMUNICATIONS/DOWN-LOADING UTILITIES</b> |

Plus over 120 other software development tools including:

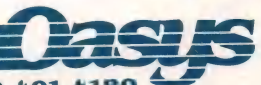
- |  |  |
|--|--|
| <ul style="list-style-type: none"> <li>• Other Complete Tool-Kits Targeting:                         <ul style="list-style-type: none"> <li>— <b>80386</b> plus 8086/186/286</li> <li>— NS32032</li> <li>— Fairchild <b>Clipper</b></li> </ul> </li> <li>• <b>C++</b> Object-Oriented C++ Translator</li> <li>• OASYS PC Platform™ 32-bit/2MB-16MB Co-Processor Board for IBM PC and Compatibles. 1-5 MIPS. UNIX and MS-DOS on the same System. Supports OASYS Tool-Kits.</li> </ul> | <ul style="list-style-type: none"> <li>• PC/Ada: Validated Ada® and APSE for IBM PC and Compatibles</li> </ul> |
|--|--|
- TOOL-KITS AVAILABLE FOR:**

VAX/VMS/ULTRIX  
SUN  
APOLLO  
GOULD  
IBM PC  
OASYS PC  
PLATFORM™  
...MANY MORE

**OASYS SERVICES:**

  - New ports easily arranged
  - OEM, site and corporate licensing
  - Training available

*Let us help you solve your puzzle.*

A DIVISION OF XEL 

60 Aberdeen Avenue, Cambridge, MA 02138 **(617) 491-4180**

Trademarks are acknowledged to: U.S. Government (AJPO), DEC, Microsoft, AT&T, and XEL, Inc.

CIRCLE 254 ON READER SERVICE CARD



# THE DOCTOR MAKES HOUSECALLS!

**Get the diagnosis from the  
Doctor in your own home.**



Subscribe to *Dr. Dobb's Journal* and enjoy the convenience of having your personal copy delivered to your home or office each month.

And you'll save over \$5 off the cover price!

Every issue of *Dr. Dobb's* will bring you indispensable programming tools like algorithms, coding tips, discussions of fundamental design issues, and actual program listings. You'll find regular coverage of:

- Popular languages such as C, Assembly, Forth, Pascal, Ada, Modula-2, BASIC, FORTRAN, and Cobol.
- 68000 and 80x86 architectures
- The MS-DOS, and Unix operating systems
- Usable techniques and practical applications of AI and object-oriented programming research.
- New product reviews, and lively discussion of professional issues in software design.
- Compilers, cross assemblers and much more!

*Dr. Dobb's Journal of Software Tools* . . . the magazine that has lived up to its reputation as the foremost source of technical tools since 1976. One year (12 information-packed issues) is just \$29.97—to subscribe simply mail in the attached card. But do it today . . . you won't want to miss *any* of the exciting issues we have planned.



DR. DOBB'S JOURNAL  
OF SOFTWARE TOOLS  
THE R & X FOR PROGRAMMERS  
SUBSCRIBE NOW AND SAVE OVER 15% OFF THE NEWSSTAND PRICE!

# SUBSCRIBE & SAVE

\$5 SAVINGS

Subscribe to **DR. DOBB'S JOURNAL** and save over \$5—a 15% savings off the cover price!

☐ Please charge my: ☐ Visa ☐ MasterCard ☐ American Express  
☐ Payment enclosed ☐ Bill me later

Card # \_\_\_\_\_ Exp. date \_\_\_\_\_  
Signature \_\_\_\_\_  
Name \_\_\_\_\_  
Address \_\_\_\_\_  
City \_\_\_\_\_ State \_\_\_\_\_ Zip \_\_\_\_\_

**ONLY \$29.97! YOU SAVE OVER \$5.00**

Savings based on a full one-year cover price of \$35.40. Canada and Mexico add \$10 for surface mail per year. All countries add \$27 for airmail per year. All foreign subscriptions must be prepaid in U.S. dollars drawn on a U.S. bank. Please allow 6-8 weeks for delivery of first issue.

A Publication of M & T Publishing, Inc.

3448

# SUBSCRIBE & SAVE

\$5 SAVINGS

Subscribe to **DR. DOBB'S JOURNAL** and save over \$5—a 15% savings off the cover price!

☐ Please charge my: ☐ Visa ☐ MasterCard ☐ American Express  
☐ Payment enclosed ☐ Bill me later

Card # \_\_\_\_\_ Exp. date \_\_\_\_\_  
Signature \_\_\_\_\_  
Name \_\_\_\_\_  
Address \_\_\_\_\_  
City \_\_\_\_\_ State \_\_\_\_\_ Zip \_\_\_\_\_

**ONLY \$29.97! YOU SAVE OVER \$5.00**

Savings based on a full one-year cover price of \$35.40. Canada and Mexico add \$10 for surface mail per year. All countries add \$27 for airmail per year. All foreign subscriptions must be prepaid in U.S. dollars drawn on a U.S. bank. Please allow 6-8 weeks for delivery of first issue.

A Publication of M & T Publishing, Inc.

3448

## COMMENTS & SUGGESTIONS

Dear Reader, June 1987, #128

Dr. Dobb's has a long tradition of listening to its readers. We like to hear when something really helps or, for that matter, bothers you. In this hectic world of ours, however, it is often difficult to take the time to write a letter. This card provides you with a quick and easy way to correspond and, if you include your name and address, we may use appropriate comments in The Letters column. Simply fill it out and drop it in the mail.

—Ed

Which articles or departments did you enjoy the most this month? Why?

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

Comments or suggestions: \_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

Name: \_\_\_\_\_  
Address: \_\_\_\_\_  
\_\_\_\_\_





No Postage  
Necessary  
If Mailed  
In The  
United States

## BUSINESS REPLY MAIL

FIRST CLASS PERMIT 790, REDWOOD CITY, CA

Postage Will Be Paid By Addressee

### Dr. Dobb's Journal of **Software Tools**

Box 3713  
Escondido, CA 92025-9843



No Postage  
Necessary  
If Mailed  
In The  
United States

## BUSINESS REPLY MAIL

FIRST CLASS PERMIT 790, REDWOOD CITY, CA

Postage Will Be Paid By Addressee

### Dr. Dobb's Journal of **Software Tools**

Box 3713  
Escondido, CA 92025-9843



PLACE  
STAMP  
HERE

### Dr. Dobb's Journal of **Software Tools**

501 GALVESTON DR.  
REDWOOD CITY, CA 94063

# DR. DOBB'S JOURNAL

## OF SOFTWARE TOOLS

THE  
**R<sub>x</sub>**  
FOR  
PROGRAMMERS

# SUBSCRIBE NOW AND SAVE OVER 15%

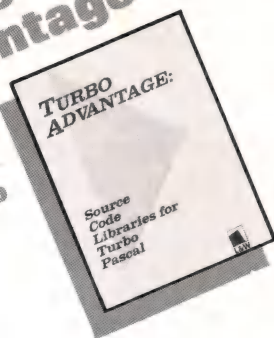
OFF THE  
NEWSSTAND  
PRICE!



# Turbo Pascal Tools

## Turbo Advantage

Source Code Libraries for Turbo Pascal



**T**his library of more than 220 routines, complete with source code, sample programs and documentation will save you hours developing and optimizing your programs!

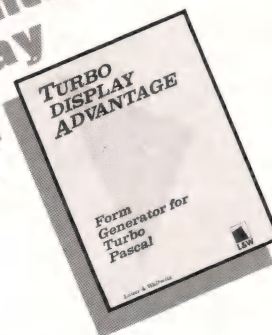
Routines are organized and documented under the following categories: bit manipulation, file management, MS-DOS support, sorting, string operations, arithmetic calculations, data compression, differential equations, Fourier analysis and synthesis, matrices and vectors, statistics, and much more! All source code is included.

A detailed manual includes a description of the routine, an explanation of the methods used, the calling sequence, and a simple example. For MS/PC-DOS systems.

**Turbo Advantage** Item #070 \$49.95

## Turbo Advantage Display

Form Generator for Turbo Pascal



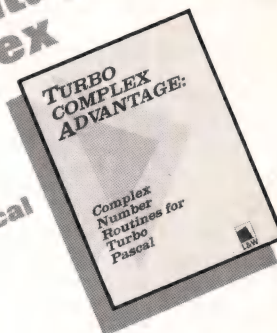
**TURBO Display** includes a menu driven form processor, 30 Turbo Pascal procedures and functions to facilitate linking created forms to your program, and full source code and documentation. For MS-DOS systems.

Some of the **TURBO Advantage: Source Code Libraries for Turbo Pascal** routines are necessary to compile **TURBO Display**.

**TURBO Display** Item #072 \$69.95

## Turbo Advantage Complex

Complex Number Routines for Turbo Pascal



**TURBO Complex** provides procedures for performing all the arithmetic operations and necessary real functions with complex numbers. Each procedure is based on predefined constants and types. By using these declarations the size of arrays are easily adapted. Each type declaration is a record with both a real and imaginary part. Use these procedures to build more sophisticated functions in your own programs.

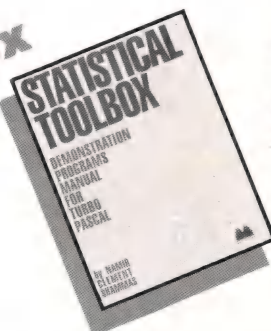
**TURBO Complex** also demonstrates the usage of these procedures in routines for vector and matrix calculation with complex numbers and variables; simultaneous Fourier transforms; calculations of convolution and correlation functions; low-pass, high-pass, band-pass and band-rejection digital filters; and solving linear boundary-value problems.

Source code and documentation is included. For MS-DOS systems. Some of the **TURBO Complex** routines are most effectively used with routines contained in **TURBO Advantage**.

**TURBO Complex** Item #071 \$89.95

## Stat Toolbox

for Turbo Pascal



Two statistical packages in one!

### A library disk and reference manual

Use these powerful statistical routines to build your applications. Routines include: • statistical distribution functions • random-number generation • basic descriptive statistics • parametric and non-parametric statistical testing • bivariate linear regression, multiple and polynomial regression.

### A demonstration disk and manual

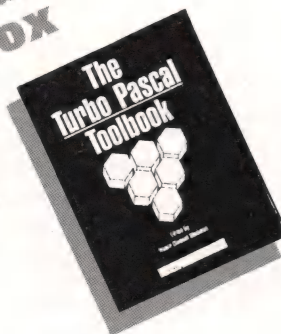
This package incorporates the library of routines into a fully functioning statistical program. Two data management programs are included to facilitate the storage and maintenance of data.

Full source code is included. (For IBM PC's and compatibles. Turbo Pascal version 2.0 or later, and PC-DOS 2.0 or later are required.)

**STAT Toolbox** Item #050 \$69.95

## Turbo Pascal Toolbox

Edited by Namir Clement Shammas



You'll find:

- an extensive library of low-level routines
- external sorting and searching tools, presenting a new database routine that combines the best features of the B-tree, B+ and B++ trees
- window management, to help you create, sort and overlay windows
- artificial intelligence techniques
- mathematical expression parsers, offering two routines that convert mathematical expressions into RPN tokens
- a smart statistical regression model that searches for the best regression model to represent a given set of data.

All routine libraries and sample programs are on disk for MS-DOS systems, and over 800K of Turbo Pascal source code is included!

**Turbo Pascal Toolbox** Item #080 \$25.95  
**Turbo Pascal Toolbox with disk** Item #081 \$45.95

**TO ORDER:** Return this coupon with your payment to: M&T Books, 501 Glaveston Dr., Redwood City, CA 94063. Or, Call TOLL-FREE 800-533-4372 Mon-Fri 8a.m.-5p.m. In CA call 800-356-2002

• **YES!** Please send me

- ☐ **TURBO Advantage** # 070 \$49.95 \_\_\_\_\_
- ☐ **TURBO Advantage Display** # 072 \$69.95 \_\_\_\_\_
- ☐ **TURBO Advantage Complex** # 071 \$89.95 \_\_\_\_\_
- ☐ **STAT Toolbox** # 050 \$69.95 \_\_\_\_\_
- ☐ **Turbo Pascal Toolbox** # 080 \$25.95 \_\_\_\_\_
- ☐ **with disk** # 081 \$45.95 \_\_\_\_\_

Subtotal \_\_\_\_\_

CA residents add saletax \_\_\_\_\_%

Add \$2.25 per item for shipping \_\_\_\_\_

**TOTAL** \_\_\_\_\_

Name \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_

State \_\_\_\_\_ Zip \_\_\_\_\_

☐ Check enclosed. Make payable to M&T Publishing.

Please charge my ☐ VISA ☐ M/C ☐ AMEX

Card no. \_\_\_\_\_

Expiration Date \_\_\_\_\_

Signature \_\_\_\_\_



## XOR CHAIN

(continued from page 35)

*Anchor*, the *Item* links, and *prior* and *next*.

### Deleting an Item

You need a way to delete an item of a list. Deleting comes down to fixing the things that point to an item so they point to other items instead. Once nothing points to it anymore, an item can be discarded. The deletion procedure in Example 2, page 36, deletes the next *Item* of the scan. If there is no next *Item*, it does nothing (ergo, it correctly does nothing to an empty list).

The item to be deleted is *s->next*. The first step is to get the address of its successor, which will take its place. It is obtained from its link field in the manner of *GoFwd*.

Next you repair the predecessor of the deleted item. If it's the head of the list, the anchor must be repaired; otherwise, the predecessor's link field is adjusted to point to the deleted item's successor.

The successor of the deleted item is then repaired. If there isn't one, the

deleted item was the tail of the list, and its predecessor becomes the new tail.

You could empty a list with this loop:

```
Associate(z,Test);
ToHead(z);
while (Not AtTail(z)) Delete(z);
```

Although this procedure won't delete when the scan is *AtTail*, the tail item can nevertheless be deleted using:

```
ToTail(z); StepBak(z); Delete(z);
```

Earlier I said that any number of *Scan* structures could be associated with a list. Now that I've defined *Insert* and *Delete*, I should perhaps modify that claim. What might the effect be on one *Scan* if *Insert* were performed via a different *Scan* of the same list? (Answer: not much—the new item might be missed by the other *Scan*.)

What about *Delete*? (Answer: catastrophe—the other *Scan* might step using the link word of a deleted, now-invalid *Item*.)

## Conclusion

The C functions shown here form the basis of a package for handling doubly linked lists. These lists have several advantages: because they are doubly linked, they can be traversed in either direction with equal ease, and insertion and deletion are simple and speedy operations. By packaging a list position as a single *Scan* structure, you make it convenient to track several list positions at once, as long as no deletions are allowed.

There are disadvantages, of course. The greatest is that the only practical access to the list is from its ends; it isn't possible to enter it at the middle and then traverse forward or backward. (Traversal requires the addresses of two logically adjacent items, which can't be maintained outside the list itself in the presence of *Insert* or *Delete*.) Thus this list form can't be indexed for quick entry at intermediate points.

Readers of an imaginative turn of mind might explore some other possibilities. For instance, the structure treated here as a linear list could be wrapped around to make an endless ring. In a double-linked ring, *Anchor* might be eliminated in favor of a mandatory single *Item*.

DDJ

Vote for your favorite feature/article.  
Circle Reader Service No. 4.

## The C Programmer's Assistant

# C TOOLSET

### UNIX-like Utilities for Managing C Source Code

No C programmer should be without their assistant—C ToolSet. All of the utility programs are tailored to support the C language, but you can modify them to work with other languages too.

Source code in standard K&R C is included; and you are welcome to use it with any compiler (UNIX compatible) and operating system you choose.

### 12 Time Savers

**DIFF** - Compares text files on a line-by-line basis; use **CMP** for byte-by-byte. Indispensable for showing changes among versions of a program under development.

**GREP** - Regular expression search. Ideal for finding a procedural call or a variable definition amid a large number of header and source files.

**FCHART** - Traces the flow of control between the large modules of a program.

**PP** (C Beautifier) - Formats C program files so they are easier to read.

**CUTIL** - A general purpose file filter.

Requires MSDOS and 12K RAM

**CCREF** - Cross references variables used within a program.

**CBC** (curly brace checker) - checks for pairing of curly braces, parens, quotes, and comments. Other utilities include **DOCMKAKE**, **ASCII**, **NOCOM**, and **PRNT**.

Source code to every program is included!

### Thorough User Support Text and Online

C ToolSet documentation contains descriptions of each program, a listing of program options (if any), and a sample run of the program.

On-line help gives you information on the programs and how to run them. Most of the programs respond to **?** on the command line with a list of options.

Call 800-821-2492 to order C ToolSet risk-free for only \$95.

**Solution Systems**™

335-D Washington St.,  
Norwell, MA 02061  
(617) 659-1571

Full refund if not  
satisfied in 30 days.

```
void Delete(s) struct Scan *s;
{
    struct Item *i;
    if (AtTail(s)) return;
    i = s->next->link
    Xor s->prior Xor s->next;

    if (AtHead(s))
        s->a->head = i;
    else
        s->prior->link Xor = (s->next
        Xor i);

    if (i == Nil)
        s->a->tail = s->prior;
    else
        i->link Xor = (s->next Xor s-
        >prior);

    DropItem(s->next);
    s->next = i;
}
```

**Example 2** A procedure to delete an item in a list

CIRCLE 152 ON READER SERVICE CARD



# FORTRAN PROGRAMMERS

LCS ANNOUNCES F77L LAHEY FORTRAN VERSION 2.20  
WE JUST MADE OUR TOP RATED FORTRAN LANGUAGE SYSTEM BETTER.

*"Lahey's F77L FORTRAN is the compiler of choice. It's definitely a 'Programmer's FORTRAN,' with features to aid both the casual and the professional programmer . . . F77L compiled the five files in a total of 12 minutes, which was 4 times as fast as MS FORTRAN and an astounding 6 times as fast as Pro FORTRAN..."*

—Editor's Choice PC Magazine

## HERE ARE JUST A FEW OF THE REASONS WHY F77L IS THE COMPILER OF CHOICE:

- Full Implementation of the ANSI 77 Standard
- Fast Compilation—outruns everything on the market
- Powerful Multi-Featured Source On-Line Debugger
- Popular Extensions for easy porting of mainframe and mini computer programs (Including NAMELIST)
- Recursion—allocates local variables on the stack
- Arrays and COMMONS greater than 64K
- Clear and Precise English Diagnostics
- Long Variable Names—Up to 31 Characters
- COMPLEX\*16, LOGICAL\*1 and INTEGER\*2
- IEEE Standard Floating Point
- Compatibility with popular third party software
- Unmatched Technical Support with an on-line bulletin board

## NEW FEATURES WITH VERSION 2.20:

- Cross reference and source listings
- Allocation maps of COMMON variables and arrays
- In-line comments
- IMPLICIT NONE compiler option
- Faster Execution
- Source On-Line Debugger (SOLD) includes:  
Trace Execution; No Relinking required;  
On-screen Listing; No effect on code size

Call about our New F77L development tools:  
Lahey Profiler Mathematical Functions Library Overlay Linker



It is more than just features that make F77L an outstanding product; it is the years of experience behind the software. At Lahey Computer Systems, we have been developing FORTRAN compilers since 1967 and we are committed to keeping F77L the industry leader.

When PC Magazine selected our version 1.35 as the Editor's Choice among PC FORTRANs, we were pleased but not completely satisfied—we knew we could improve the product. F77L Version 2.20 increases our lead over the competition. F77L's precise diagnostics, advanced debug package, helpful user screens and comprehensive manual make it a complete and easy to use high productivity tool.

When evaluating any software package, an important factor to consider is the value of your time. F77L saves you time and

money the moment you start using it. Our FORTRAN Language System has the key features you need to increase productivity and get the job done. Other PC FORTRANs may be cheaper than F77L, but none are less expensive to use.

## F77L—THE PROGRAMMER'S FORTRAN

Price: \$477.00

System Requirements: MS-DOS or PC-DOS (2.0 or greater),  
256K, math coprocessor (8087-80287)

TO ORDER OR FOR MORE INFORMATION:  
**702-831-2500**



Lahey Computer Systems, Inc.  
P.O. Box 6091, Incline Village, NV 89450  
Telex: 9102401256

## International Representatives:

Canada: Barry Mooney & Assoc., Tel. (902)6652941 • England: Grey Matter Ltd. Tel. (0364)53499 • Switzerland: DST Comp. Services, Tel. (022)989188  
Denmark: Ravenholm Computing, Tel. (02)887249 • Australia: Comp. Transitions, Tel. (03)5372786 • Japan: Microsoft Inc., Tel. (03)8138222

MS-DOS & MS FORTRAN are trademarks of Microsoft Corporation. Pro FORTRAN refers to IBM PC Professional FORTRAN by Ryan McFarland



# THE PROGRAMMER'S SHOP

helps save time, money and cut frustrations. Compare, evaluate, and find products.

## RECENT DISCOVERY

**UI Programmer** - Quickly generate dBASE User Interfaces, prototypes. Use supplied templates or create own. Pop-up help, bounce bar menus, screen forms. II, III, FoxBASE+, Quicksilver, Clipper. PC \$ 295

## AI-Expert System Dev't

Arity Combination Package PC \$ 979  
System - use with C MS \$ 229  
SQL Dev't Package MS \$ 229  
Auto-Intelligence PC \$ 749  
Experteach - Powerful, samples PC \$ 349  
Exsys PC \$ 309  
Runtime System PC \$ 469  
Insight 2+ MS \$ 379  
Intelligence/Compiler PC \$ 749  
T.I. - PC Easy PC \$ 435  
Personal Consultant Plus PC \$2589  
Personal Consultant Runtime PC \$ 85  
Turbo Expert-Startup-(400 rules) PC \$ 129  
Corporate (4000 rules) PC \$ 359

## AI-Lisp

Microsoft MuLisp 85 MS \$ 159  
PC Scheme LISP - by TI PC \$ 85  
TransLISP - learn fast MS \$ 89  
TransLISP PLUS  
Optional Unlimited Runtime MS \$ 150  
PLUS for MSDOS MS \$ 179  
Others: IQ LISP (\$239), IQC LISP (\$269)

## AI Prolog

APT - Active Prolog Tutor - build applications interactively PC \$ 65  
ARITY Prolog - full, 4 Meg  
Interpreter - debug, C, ASM PC \$ 229  
COMPILER/Interpreter-EXE PC \$ 569  
Standard Prolog MS \$ 77  
MacProlog Complete MAC \$ 269  
MicroProlog - Prof. Entry Level MS \$ 85  
MicroProlog Prof. Comp./Interp. MS \$ 439  
MPROLOG P550 PC \$ 175  
Prolog-86 - Learn Fast MS \$ 89  
Prolog-86 Plus - Develop MS \$ 229  
TURBO PROLOG by Borland PC \$ 69

## Editors for Programming

BRIEF Programmer's Editor PC Call  
EMACS by UniPress Source: \$929 \$ 299  
Epsilon - like EMACS, full  
C-like language for macros. PC \$ 149  
KEDIT - like XEDIT PC \$ 99  
Lattice Screen Editor - multiwindow, multitasking Amiga \$ 89 MS \$ 109  
Micro Focus Micro/SPF PC \$ 139  
PC/EDT - macros PC \$ 229  
PC/VI - by Custom Software MS \$ 109  
Personal REXX PC \$ 99  
PMATE - power, multitask PC \$ 119

Note: All prices subject to change without notice.  
Mention this ad. Some prices are specials. Ask about COD and POs. Formats: 3" laptop now available, plus 200 others.  
UPS surface shipping add \$3/item.

## Compare Products Use Product Specialists

The Programmer's Shop is much more than the supplier of the largest selection of programmer's software. Trained programming consultants will answer your questions. Ask "product specialists" about Cross Assembler, Translators, Debuggers, or compilers.

### Our Services:

- Programmer's Referral List
- Compare Products
- Help find a Publisher
- Evaluation Literature FREE
- BBS - 7 PM to 7 AM 617-826-4086
- Dealers Inquire
- Newsletter
- Rush Order
- Over 700 products
- National Accounts Center

## C Language-Compilers

AZTEC C86 - Commercial PC \$499  
C86 PLUS - by CI MS Call  
Datalight C - fast compile, good code, 4 models, Lattice compatible, Lib source. Dev's Kit PC \$ 77  
Datalight Optimum - C MS \$ 99  
with Light Tools by Blaise PC \$168  
Lattice C - from Lattice MS \$275  
Let's C Combo Pack PC \$ 99  
Let's C PC \$ 59  
Microsoft C 4.0- Codeview MS \$275  
Uniware 68000/10/20 Cross Compiler MS Call  
Rex - C/86 by Systems & Software - standalone ROM MS \$695  
Wizard C MS \$299  
Rom Development Package MS \$259

## C Language-Interpreters

C-terp by Gimpel - full K & R MS \$229  
C Trainer - by Catalytix PC \$ 89  
INSTANT C - Source debug, Edit to Run-3 seconds, .OBJS MS \$379  
Interactive C by IMPACC Assoc. PC \$209  
Run/C Professional MS \$159  
Run/C Lite MS \$ 89

## C Libraries-General

Blackstar C Function Library PC \$ 79  
C Essentials - 200 functions PC \$ 75  
C Function Library MS \$109  
C Tools Plus (1 & 2) - Blaise PC \$125  
C Utilities by Essential PC \$129  
C Worthy Library - Complete, machine independent MS \$249  
Entelekon C Function Library PC \$119  
Entelekon Superfonts for C PC \$ 45  
Greenleaf Functions-portable, ASM \$139  
LIGHT TOOLS by Blaise PC \$ 69

## FEATURE

**Personal COBOL** by MicroFocus - Develop, test, debug, execute ANSI '74 code. Full-screen editor, syntax checker, Animator, forms/screen generator, help. Compatible with Level II. PC \$169

## RECENT DISCOVERY

**NET-TOOLS** - Access NETBIOS-compatible network systems from Microsoft C, Pascal, FORTRAN, Assembler, Lattice C. Full Source, No Royalties. PC \$129

## dBASE Language

Clipper compiler PC Call  
dBASE II MS \$329  
dBase III Plus PC \$429  
dBASE III LanPack PC \$649  
DBXL Interpreter by Word Tech PC \$139  
FoxBASE+ - single user MS \$349  
Quick Silver by Word Tech PC \$499

## dBASE Support

dBase Tools for C PC \$ 65  
dBrief with Brief PC Call  
DBC ISAM by Lattice MS Call  
dBx Translator to C MS \$319  
dFlow - flowchart, xref MS Call  
Documentor - dFlow superset MS Call  
Genifer by Bytel-code generator MS \$299  
QuickCode III Plus MS \$249

## Fortran & Supporting

50:More FORTRAN PC \$ 99  
ACS Time Series MS \$399  
Forlib+ by Alpha MS \$ 59  
MS Fortran - 4.0, full '77 MS \$279  
No Limit - Fortran Scientific PC \$115  
PC-Fortran Tools - xref, pprint PC \$179  
RM/Fortran MS Call  
Scientific Subroutines - Matrix MS \$139



## Multilanguage Support

BTRIEVE ISAM MS \$185  
BTRIEVE/N-multiuser MS \$455  
Flash-Up Windows PC \$ 79  
GSS Graphics Dev't Toolkit PC \$375  
HALO Graphics PC \$209  
Development Package MS \$395  
Informix - by RDS PC \$639  
Informix 4GL-application builder PC \$789  
Informix SQL - ANSI standard PC \$639  
Opt Tech Sort - sort, merge MS \$ 99  
PANEL MS \$215  
Pfinish - by Phoenix MS \$229  
PolyLibrarian by Polytron MS \$ 79  
PolyBoost - speed I/O, keyboard PC \$ 69  
PVCS Corporate-source control MS \$319  
QMake by Quilt Co. MS \$ 85  
Report Option - for Xtrieve MS \$109  
Screen Sculptor PC \$ 95  
SRMS - source control MS \$109  
VXM - multi-env. link MS \$195  
Xtrieve - organize database MS \$199  
ZAP Communications - VT 100 PC \$ 89

## FEATURE

**Turbo C** by Borland. ANSI Compiler supports 6 models including tiny and huge, has floating point, interactive editor, Make. Speed development. PC \$ 69

We support MSDOS (not just compatibles), PC DOS, Xenix-86, CPM-80, Macintosh, Atari ST, and Amiga.

 **HOURS**   
8:30 AM-8:00 PM EST.

**Call for a catalog, literature,  
advice and service you can trust**

"I have been pleased with your catalog selection, the knowledge of your telephone answering staff, and the promptness of your service. I have every wish to become a continuing customer of The Programmer's Shop."

Carl C. Rollo

CIRCLE 133 ON READER SERVICE CARD



# THE PROGRAMMER'S SHOP

provides complete information, advice, guarantees and every product for Microcomputer Programming.

## AUTO-INTELLIGENCE

### The First Automatic Knowledge Acquisition System

List: \$990

Our: \$749

**IntelligenceWare, Inc.**

9800 S. Sepulveda Blvd. Suite 730

Los Angeles, CA 90045

Telephone: (213) 417-8896; Fax (213) 417-8897

### WANT TO ADD WINDOWS, ICONS, FONTS, FAST GRAPHICS, DIALOG BOXES, PROCESS MANAGEMENT, AND DEVICE INDEPENDENCE

TO YOUR IBM PC PROGRAMS?

If you have ever wished that you could develop stunning Macintosh-like programs on the IBM PC without the overhead of an enormous operating environment like Windows or GEM, then you need the **SYNERGY DEVELOPMENT TOOLKIT**, from **Matrix Software**.

Using a memory resident runtime module only 20K in size (versus as much as 300K for Windows), you can develop applications with: overlapped and tiled windows; pull-down menus with half intensity options and automatic sizing; fast graphics function calls to draw shapes, lines, boxes, and create intricate fill patterns in both regular and irregular areas; have full device independence (drivers for numerous devices, including CGA, EGA and Hercules are included); sophisticated text input and output, with fonts in different styles and sizes; full keyboard support (your programs won't need a mouse!) and powerful mouse support; and process management calls to efficiently manipulate system resources.

The Toolkit has gateways to support the following languages:

- |                  |                         |                           |
|------------------|-------------------------|---------------------------|
| • Turbo Pascal   | • Microsoft & Lattice C | • Basic                   |
| • IBM/MSP Pascal | • Macro Assembler       | • dBASE III/III Compilers |

In addition, the Toolkit includes a powerful collection of tools including a graphics resource editor for developing your own icons and fonts.

**NEW!** The Toolkit also includes a free copy of **Synergy Layout**, a revolutionary software development tool that dramatically increases your productivity by actually generating bug-free source code in Macro, C, and Turbo Pascal.

For further information, contact Matrix Software at (617) 567-0037.

List: \$395

Our: \$349

CIRCLE 302 ON READER SERVICE CARD

## MUMPS

Now quickly and easily develop  
multi-tasking applications on your PC!

Now get the speed, power and flexibility of MUMPS in a truly concurrent processing environment with COMP Computing Standard MUMPS (CCSM).

CCSM provides the programmer with these benefits:

- Save time and money by writing Applications in 1/3 to 1/5 the amount of code.
- Fast data access with B-Tree File Structure and Automatic disk caching.
- Unlimited program size and variable space with advanced virtual memory system.
- 100% portable from micros to minis to mainframes.
- Easily write multi-tasking and multiuser applications.

And with MUMPS, all you need to do to start a background process is: "JOB ^ROUTINE". No memory allocation, no table set-up, no hassles; the system handles everything.

CCSM is also available in a multi-tasking version for the Macintosh.

Multi-Tasking:	List - \$149.95	Ours - \$129
Multiuser:	List - \$450.00	Ours - \$369

For a demonstration diskette of CCSM, send two dollars to the address below.

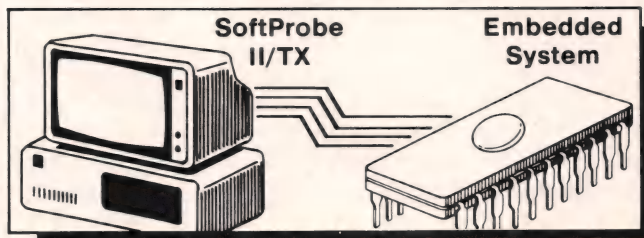
**MGlobal**

1601 Westheimer, Suite 201  
Houston, Texas 77006

**1-800-257-8052**

In Texas: 1-713-529-2576

## SoftProbe II/TX, a High-Level Solution for Embedded System Debugging.



### FEATURES

- Source Level Debug
- Supports High Level Language Data Types
- C-like Command Language with Expression Evaluation and Flow Control
- Displays Program Execution Trace in High Level and Assembly Languages
- Friendly User Interface with Macros and On-line Help

### TARGET SYSTEM

- Any iAPX-86/186 Based System with a USART
- IBM PC or Compatible.

### BENEFITS

- Debug on Actual Target System
- Debug at High Level Language Level
- Mixed Language Debugging Facilitates Firmware Testing

### LANGUAGE SUPPORT

- C • PL/M • ASM

**SYSTEMS & SOFTWARE**

3303 Harbor Blvd.,  
C-11, Costa Mesa, CA 92626  
(714) 241-8650 FAX (714) 241-0377

List: \$750 Ours: \$695

SoftProbe is a registered trademark of Systems & Software, Inc.  
IBM PC is a registered trademark of International Business Machines Corp.

Call Today for FREE detailed  
information or try Risk-Free for 31 days.

**800-421-8006**  
HOURS: 8:30 A.M. - 8:00 P.M. E.S.T.

**THE PROGRAMMER'S SHOP™**  
Your complete source for software, services and answers  
5-DPond Park Road, Hingham, MA 02043  
Mass: 800-442-8070 or 617-740-2510 2/87



# An Extended IBM PC COM Port Driver

by Thomas A. Zimniewicz

**T**he IBM PC and its compatibles support two serial ports—COM1 and COM2—in a wide variety of configurations. Four basic functions are provided—data in, data out, set configuration, and read status. What the IBM PC does not do is walk and chew gum at the same time. If your disk is seeking and more than one character arrives, it is lost. This article describes what I think a COM port driver should do, discusses some design rules and implementation pitfalls of device drivers, and presents a device driver modification with the following features:

- buffered input—no lost data
- input flow control—XON/XOFF or control lines
- output flow control—control lines
- works with existing software
- removal—without a reboot
- 19,200- and 38,400-baud operation
- type-ahead with CRTs
- upward compatible

## What's Wrong?

During the first part of my career as a software engineer, I was spoiled by minicomputer operating systems that were good at capturing serial data. More recently, I had the pleasure of working with Unix—a pleasure, that is, in all aspects except capturing serial data. My introduction to Unix was on a PDP-11/44 that could miss characters from serial data input at 300 baud on a busy day. Every time my job required monitoring serial data, it was a bad day. Then one

***What the IBM PC  
does not do  
is walk and chew gum  
at the same time.***

day I noticed an idle IBM PC that had nothing to do (no other users to get in my way) except perhaps collect my data. It would be simple—just *getchar* from COM1 and *putchar* to disk, and with no other work, the IBM PC could certainly handle at least 9,600 baud. A while later it was working, but (and there always is one) it seemed that there were gaps in the data. (I discovered later that the gaps coincided with the flashing of the disk light.) I gave up. Much later I found myself using an IBM PC a little. I didn't like the keyboard and monitor, so I hooked up my CRT to a COM port and then found there was no type-ahead. I got mad enough to fix it.

So, what were my goals? The main thing was buffered interrupt-driven input to allow type-ahead and high-speed serial data capture. I wanted to use these features with existing programs, too. This meant the solution had to be fully compatible and somehow get between programs and the hardware. I had to modify PC-DOS and/or the BIOS in a way that was transparent to an application-level program. In this sea of bad news, there was one very bright light—everything that PC-DOS and the BIOS lacked in capability, it made up for in flexibility. The hooks were there, device drivers could be installed, inter-

rupts could be stolen. If the new COM port software were installed on an IBM PC, then all well-behaved programs (those that used PC-DOS or the BIOS instead of directly accessing the hardware) would automatically gain the new features. Several methods of input/output flow control were possible and the hardware could handle higher baud rates . . . I sure do get carried away easily.

## Can It Be Fixed?

The IBM PC actually has two levels of software to handle the COM ports. The high-level code is part of PC-DOS and is loaded into RAM during the boot-up of PC-DOS. This code handles those functions that do not depend on the details of the hardware and provides an interface to the COM port. This level is called a device driver and is accessed by high-level PC-DOS operations such as those that open and read serial data system calls. The low-level code is hardware specific and is contained in ROM. This code is specific to the operation of the COM port hardware. This level is called the BIOS and is called by the device driver to perform operations specific to a COM port, such as setting the baud rate or getting a character from the port hardware.

When I started to investigate the job at hand, I suspected that a whole new device driver would be required. The good news was that PC-DOS faithfully used the BIOS for all COM port operations. This meant that all I needed to do was to write a replacement for the parts of the BIOS that handle the COM port. For a while the job looked really easy. I considered stealing the software interrupt

*Thomas A. Zimniewicz, 2695 Pond Rd., Lima, NY 14485. Thomas is a software consultant.*



(*int 14h*) that is used to access the BIOS COM port code, going to my code for operations that needed changing, and otherwise just passing the job off to the BIOS. Unfortunately, I had to abandon this easy way out because of negative side effects caused by almost every BIOS function. For example, the BIOS "send a character" function sets the control signals in ways that are incompatible with some of the required flow control modes.

To illustrate some of the technical issues of handling a COM port, let's take a simplified look at the job of reading serial data. When the BIOS call is made to read a character, it loops until a character arrives at the port or a counter expires. Either the character or a time-out status is returned. Most programs and PC-DOS ignore the time-out and try again. The COM port hardware has only a single character buffer. If the buffer in the COM port hardware overflows while the processor is doing anything else, it is lost. The solution to these lost characters is to have the COM port generate an interrupt when a character arrives and have the interrupt handler save the character in a buffer until the program gets around to needing it. The code might look like this:

```
COM port interrupt handler
  read character
  if buffer count < buffer size
    put character in the buffer
    increment buffer count
  else
    set overrun error

read character routine
  loop
    if buffer count > 0
      decrement buffer count
      take character and return it
    if timer expired
      return time-out status
  endloop
```

If this algorithm were implemented, it would probably pass all its initial tests. Then, when you least expected it, your input would be garbled and no errors would be reported. What you have here is your classic race condition. Imagine the buffer is exactly full when the read routine is called and the count is decremented to indicate room in the

buffer. Immediately after the count is decremented but before the character is removed, an interrupt occurs. The handler puts another character in the buffer that is really still full. Things go downhill from here. This type of error is an easy trap for beginners (and sometimes a big embarrassment for pros). One possible fix is to disable interrupts during critical portions of the read character routine. (Note that COM port interrupts are automatically disabled by the hardware during the processing of a COM port interrupt; there is no need to worry about reentering the handler itself). This works, but it has

### ***I wanted buffered input to allow type-ahead and high-speed serial data capture.***

serious drawbacks. If interrupts are off for too long, serial data (and other real-time events) can be lost. The best solution is to order the events in the read character routine so that an interrupt cannot hurt it. If interrupts must be disabled, try to limit the duration. To fix the bug above, just take the character from the buffer before decrementing the buffer count.

#### ***Code Walk-Through***

Excom is functionally equivalent to the BIOS *int 14h* COM port handler with three significant enhancements—it provides interrupt-driven buffered input and has an extended set of configuration parameters, including flow-control selection and higher baud rates. Excom is made up of three major sections—the COM port data input interrupt handler, the replacement for the BIOS *int 14h* handler, and the code to install and initialize excom. A problem exists in trying to describe excom, unlike a well-written application in which functions can be isolated to a single routine or group of routines, because its structure dictates that many of its capabilities are distributed through-

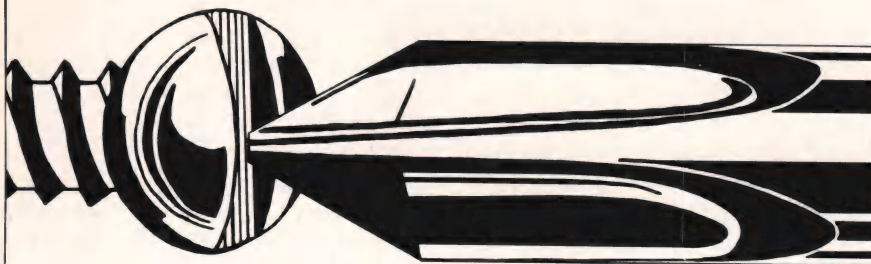
out the code. I will first take you on a quick trip through the listing (Listing One, page 64) to show its structure and then later describe several of excom's more interesting functions.

Excom is a .COM program that is run once before the COM ports are needed and that uses the terminate-and-stay-resident feature of PC-DOS to remain active as long as is needed. If required it can be removed, restoring the interrupt vectors to their previous values and freeing memory. Two requirements for a .COM program are to have all segment registers pointing to the start of the program's memory space and to have an executable origin of *100h* to make room for the program header. Execution starts at location *100h*, which is a jump to the initialization code. A few constants are then defined, the first three of which have to do with the buffer size and can be modified to tune the excom buffers to your needs. The Microsoft assembler supports a concept similar to structures in C. A structure called *pcb* (port control block) is defined to describe the dynamic state of a port. It is defined as a structure so that one piece of code can handle two ports just by setting a register to point to the appropriate structure. Storage for two port control blocks is then allocated. Space is reserved to store the old interrupt vectors required for the program to remove itself. Next the baud rate table contains "magic" numbers required to set the UART's baud rate clocks. Note that the last two entries are used for the excom extended baud rates.

The first executable code is the COM port hardware interrupt handler. The label *int0B* is the entry point for *int 0Bh*, COM2, whereas *int0C* is for *int 0Ch*, COM1. Each entry sets *ds:si* to point to the *pcb* associated with its port and enters common code. The rest of the routine reads the character, puts it in the buffer, takes any action required for flow control, and exits. There is no analogy to this code in the BIOS. The second code segment is the *int 14h* handler, which performs five distinct functions corresponding to the value in the register *ah*. The functions are initialize port, transmit a character, receive a character, get port status, and extend initialization. At the



# ISN'T IT A PITY...



## Everything Isn't As Accommodating As

**c-tree**<sup>TM</sup> / **r-tree**<sup>TM</sup>  
FILE HANDLER / REPORT GENERATOR

### Performance and Portability

For all the time you devote to developing your new programs, doesn't it make sense to insure they perform like lightning and can be ported with ease?

### c-tree: Multi-Key ISAM Functions For Single User, Network, & Multi Tasking Systems

Based on the most advanced B+ Tree routines available today, **c-tree** gives you unmatched keyed file accessing performance and complete C Source Code. Thousands of professional C programmers are already enjoying **c-tree**'s royalty-free benefits, outstanding performance, and unparalleled portability.

Only FairCom provides single and multi-user capabilities in one source code package, including locking routines for Unix, Xenix, and DOS 3.1, for one low price! In addition, **c-tree** supports fixed and variable record length data files; fixed and variable length key values with key compression; multiple indices in a single index file; and automatic sharing of file descriptors.

### r-tree: Multi-File Report Generator

**r-tree** builds on the power of **c-tree** to provide sophisticated, multi-line reports. Information spanning multiple files may be used for display purposes or to direct record selection. You can develop new reports or change existing reports without programming or recompiling and can use any text editor to

create or modify **r-tree** report scripts including the complete report layout. At your option, end users may even modify the report scripts you provide.

### Unlimited Virtual Fields; Automatic File Traversal

**r-tree** report scripts can define any number of virtual fields based on complex computational expressions involving application defined data objects and other virtual fields. In addition, **r-tree** automatically computes values based on the MAX, MIN, SUM, FRQ, or AVG of values spread over multiple records. **r-tree** even lets you nest these computational functions, causing files from different logical levels to be automatically traversed.

Unlike other report generators, **r-tree** allows you to distribute executable code capable of producing new reports or changing existing reports without royalty payments, provided the code is tied to an application. Your complete source code also includes the report script interpreter and compiler.

### How To Order

Put FairCom leadership in programmers utilities to work for you. Order **c-tree** today for \$395 or **r-tree** for \$295. (When ordered together, **r-tree** is only \$255). For VISA, MasterCard and C.O.D. orders, call 314/445-6833. For **c-tree** benchmark comparisons, write FairCom, 2606 Johnson Drive, Columbia, MO 65203.



Complete C Source Code & No Royalties!

Xenix is a registered trademark of Microsoft Corp. Unix is a registered trademark of AT&T

CIRCLE 93 ON READER SERVICE CARD

### COM PORT DRIVER (continued from page 43)

end of excom is the initialization. This is the code that runs only when excom is being installed. Initialization starts by releasing the memory for excom's copy of the environment, which is not used. The body of the initialization uses PC-DOS calls to get the old interrupt vectors and then install the three new interrupt handlers. The interrupts that are thus stolen are 0Bh (COM1 hardware), 0Ch (COM2 hardware), and 14h (BIOS COM port calls). Then the excom COM port data structures are initialized. This initialization code is at the end so that it is not kept as part of the resident code. Initialization ends with the PC-DOS call to make everything before the initialization code stay resident.

### Key Functions

The BIOS does COM port input by waiting for a character to arrive. Even though it is not used, the IBM PC has all the hardware required to generate an interrupt when a character arrives. This capability is activated by changing the interrupt vector (a RAM location) to point to the appropriate handler (see lines 555-592 of Listing One). Now the vectors are set up, but because the BIOS doesn't use COM port interrupts, the hardware has not been initialized to generate interrupts. Two separate pieces of hardware have to be initialized—first the COM port hardware (an 8250 UART) has to be programmed to generate interrupts and then the interrupt controller (an 8259) has to be set to allow the interrupt to be sent on to the CPU. I expected this to be a simple flip through the data sheets and a dozen lines of code. It didn't work—no interrupts were generated. After much pain, I discovered that a spare output called OUT2 on the UART was used to gate the interrupt signal from the UART. I have no idea why. I then asserted OUT2 on the UART and voilà—interrupts (see lines 290-295). When an interrupt does occur, the CPU stops what it's doing and jumps through the appropriate vector—in this case to int0B or int0C (lines 73-158). The interrupt handler then reads the character, which resets the interrupt status in the UART and



sends an interrupt complete command to the interrupt controller before returning.

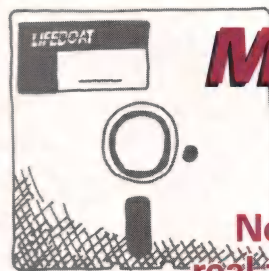
The BIOS port status request simply reads the two UART status registers and returns them to the caller. One register is called the line status and contains bits for receive errors, receive data ready, time-out, and transmit ready. The other is called modem status and has the UART's control lines. When characters are transmitted or received, the *ah* register returns the line status. When a port is initialized or status is requested, *ah* is returned as above and *al* returns the modem status. In *excom* input characters are stored in a circular buffer as they are read from the UART and are then removed from the buffer when a character read takes place. This presents a dilemma for *excom* because some status bits are associated with the received character whereas others are really UART status. The solution is to have the interrupt handler read the status associated with receive characters and buffer both character and status together. Then when a status is needed and there are characters in the buffer, the status associated with input characters is taken from the next character in the buffer. The rest of the status is taken directly from the UART (see lines 429-457). This approach is not always perfect, but *excom* has been used successfully with many port handling packages, including *COMMAND.COM*.

One of the major goals for *excom* was to provide configurable flow control for input data. The objective of flow control is to signal the source of the data to stop sending when the buffer is getting full (see lines 96-120). Then, when the buffer is emptied a bit, the sender is told to resume sending (see lines 394-409). The exact number of characters in the buffer when input is stopped and started is configurable (lines 16-18). Having the stop and restart values be different by a few characters tends to minimize the number of times the data is stopped and restarted. The actual mechanisms used to tell the sender to stop are any combination of DTR (data transfer register), RTS (request to send), and XON/XOFF (Control-S/Control-Q). DTR and RTS are turned off as the buffer fills and turned back

on when room becomes available. Control-S is transmitted as the buffer fills, and Control-Q is transmitted when room becomes available. I always thought that RTS was used as described by its name, but I've seen others use it for input flow control. It was easy to add and it didn't seem to do any harm, so I used it for input flow control, too. Add to this arsenal the ability to cross a few wires in a cable, and almost any situation is covered—except, of course, the sender that ignores your attempts. This is where *excom* shines—interrupts and big buffers provide great power. Data can be copied from a COM port to a floppy on a lowly old 4.77-MHz IBM PC at 9,600 baud without missing a character. To complete the picture, I also added output flow control. The BIOS requires that both DSR and CTS be set before a character can be sent. *Excom* can be configured to ignore either or both of these signals.

In order to be useful, *excom* had to be upward compatible with the BIOS *int 14h*. The area of extended initialization options presented some special problems. There are no unused

bits in the BIOS set configuration command. Therefore I had to add a new command—*int 14h* with *ah* having the values from 0 to 3 were already used, so *ah* = 4 was the obvious choice (lines 460-499). The problem was that the old initialization—*ah* = 0—included the baud rate and the new initialization could also specify baud rates not included in the BIOS. The obvious solution was to require that the BIOS initialization occur before the extended initialization. If *excom* was to be used with existing software, though, this restriction was impractical. The actual solution was to have *excom* ignore the baud rate in the BIOS initialization if the extended initialization had been used to set a nonstandard baud rate (lines 304-306). Thus, you set the baud rate to 19,200, start your existing software, and a BIOS initialization call is made, but *excom* ignores the baud rate. Another concession to upward compatibility is that the extended options are relative to the BIOS defaults. *Excom* allows DTR input flow control to be enabled, whereas output CTS flow control can be disabled.



## MULTITASKING with TimeSlicer

Now, create multithreaded and  
real-time MS-DOS applications in C.

*TimeSlicer* is a linkable library of C functions to create multitasking and real-time programs at the application level rather than interfacing with the operating system.

- No limit to number of tasks that can be run concurrently.
- Tasks can be created, suspended or terminated at run-time.
- Highly efficient—10,000 context switches/second; 80 micro seconds interrupt latency.
- Supports large and small memory models; preemptive and non-preemptive modes; waking up of tasks to optimize special event processing; and interrupt service routines written entirely in C.
- Extensive intertask communication capability.
- Optimizes processor usage transparently.
- Includes examples programs with source code.
- Compatible with Lattice C, Microsoft C, ADVANTAGE C++ and assembly language.



To order or obtain a complete  
technical specification sheet call:

**1-800-847-7078**

In NY: 914-332-1875

55 South Broadway Tarrytown, NY 10591

# LIFEBOAT

The Full-Service Source for Programming Software

CIRCLE 118 ON READER SERVICE CARD



### **Not All Is Gold**

Along the way from the problem to the solution, there were many little struggles, one of which may be of interest. The rest are too embarrassing to put into print.

While first using excom, I ran into a strange problem. I have a home-made shell that reads the keyboard (CRT hooked to COM port) by calling *int 14h* directly. I ran a program that did CRT output via PC-DOS but did not read any keyboard characters. While it was running, I typed a command ahead. When the program stopped, my type-ahead was echoed minus the first character. This was early in my use of excom, so I assumed a bug in excom. Ignoring the problem, I continued what I was doing. Some time later I ran an interactive program that used PC-DOS for keyboard inputs, and my lost character from 20 minutes ago appeared on the screen as the first character echoed. Wow! At first I passed it off as a bug in excom's input buffer and a coincidence. It happened again! I considered a career as a crossing guard.

You may have noticed that I described the exact way in which the above programs read their input—that was the key to understanding. PC-DOS makes a feeble attempt to do some input buffering, and when PC-DOS calls *int 14h* to transmit a character to the port, a status is returned. If a data ready is indicated, PC-DOS does a data read and saves the character for later. PC-DOS only stores one character in this way as far as I can tell. So my type-ahead was eaten by PC-DOS, to be provided to the next call to PC-DOS for a character. Beware of mixing PC-DOS and BIOS character reads!

### **Anything Left to Do?**

Of course there is! The two questions I've heard most often as a software engineer are "When will the code be complete?" and "How many more bugs are there?" The answers have always been the same: "About a week or two" and "Three." The answers are the same here.

If you put a file on a floppy and enter the command *type file*, you will notice a flip-flop of data scrolling by and disk activity. This is because

the BIOS waits for both the disk and the display. Excom has input buffering to prevent loss of data. Output buffering won't fix any problems, but it will speed up data output in the case when the source of the data is a slow device. If excom had output buffering, it would take the characters faster than they were actually being transmitted. Thus PC-DOS would think a block of data had been sent and go to the source for more while excom was still sending data out of its buffer. The flow would be limited by making PC-DOS wait when excom's output buffers were full.

Earlier I mentioned that excom

---

## **An interesting possibility with excom is nesting several invocations.**

---

can work with any well-behaved program. What about the naughty ones? At work, I use an old version of CrossTalk that steals the COM port interrupt and doesn't put it back. I don't know if this has been fixed. Are you listening CrossTalk folks? Upon return from CrossTalk, excom has lost the COM port interrupt. One possible solution would be to run CrossTalk from a batch file that removed excom, ran CrossTalk, and then reinstalled excom. The problem with this is that you must identify and fix all offenders. It may be possible for excom to examine its environment for damage and perform repairs. I'm sure this would not cover all such problems, but I think all of them that I am currently aware of could be fixed. Self-repairing programs—maybe I should do an article on "Core Wars."

When I use my shell on a COM port with or without excom installed, XON/XOFF output flow control seems to work. There must be problems. I have heard of requests to provide XON/XOFF in relation to printer driv-

ers and so on. In any case, because PC-DOS may not see characters until some time after excom does, any PC-DOS attempt at XON/XOFF output flow control is hampered by excom. This implies that XON/XOFF output flow control should be added to excom.

Because the IBM PC does not have memory management hardware, having a memory-resident program occupy the wrong location can cause problems. When a shell runs a program, it is loaded just after the shell. If the program remains resident upon its termination, the shell is in a squeeze. The shell cannot get more memory to hold shell variables, environment, temporary buffers, and so on. A desirable extension to excom would be to have it relocate itself to the end of RAM and remain resident there.

Excom has a bug! Software is no fun if its perfect. Control-S and Control-Q are sent without any regard to the UART being ready or the control lines being correct. This sounds pretty bad, but in reality it's only a problem when there is a heavy flow of data in both directions, which is pretty rare. The reason I allow this bug to exist is that the proper solution requires interrupt-driven output. *DDJ* has only so many pages for my giant listings.

The version of excom on the listings disk is an updated version with interrupt-driven output, corrected sending of Control-S/Control-Q, the ability to repair the damage done by ill-behaved programs, the ability to do output XON/XOFF flow control, and self-relocation to the end of RAM. (See the "Availability" section at the end of this article.)

### **Use Excom on Your IBM PC**

To make excom.com, use the following commands:

```
masm excom.asm ;
link excom.obj ;
exe2bin excom.exe excom.com
del excom.exe
```

Normally, you should install excom early in the boot process—before a *print /d*—because the print spooler steals *int 14h*. The real-time clock initialization program on my clone messes up the *int 14h* vector for an unknown reason; therefore I in-



# "How to protect your software by letting people copy it"

By Dick Erett, President of Software Security



Inventor and entrepreneur, Dick Erett, explains his company's view on the protection of intellectual property.

**"A** crucial point that even sophisticated software development companies and the trade press seem to be missing or ignoring is this:

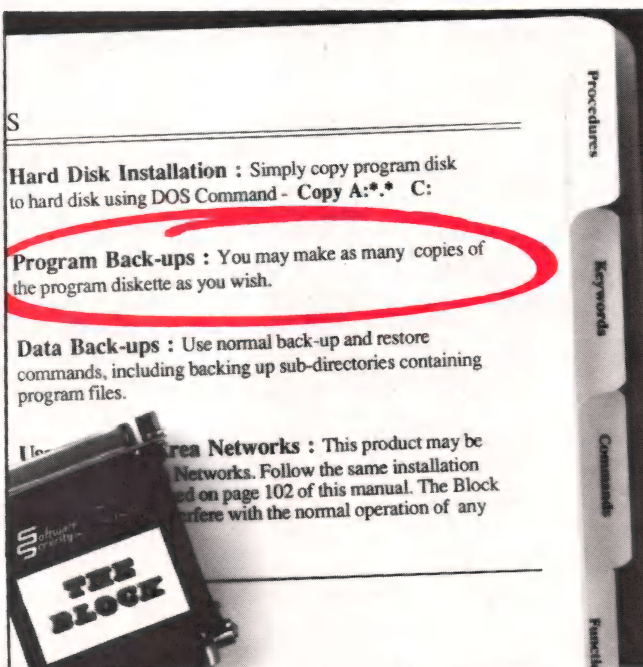
*Software protection must be understood to be a distinctively different concept from that commonly referred to as copy protection.*

Fundamentally, software protection involves devising a method that prevents unauthorized use of a program, without restricting a legitimate user from making any number of additional copies or preventing program operation via hard disk or LANs.

Logic dictates that magnetic media can no more protect itself from misuse than a padlock can lock itself.

Software protection must reside outside the actual storage media. The technique can then be made as tamper proof as deemed necessary. If one is clever enough, patent law can be brought to bear on the method.

Software protection is at a crossroads and the choices are clear. You can give product away to a segment



*Soon all software installation procedures will be as straightforward as this. The only difference will be whether you include the option to steal your product or not.*

of the market, or take a stand against the theft of your intellectual property.

*"...giving your software away is fine..."*

We strongly believe that giving your software away is fine, if you make the decision to do so. However, if the public's sense of ethics is determining company policy, then you are no longer in control.

We have patented a device that protects your software while allowing unlimited archival copies and uninhibited use of hard disks and LANs. The name of this product is The BLOCK™.

The BLOCK is the only patented method we know of to protect your investment. It answers all the complaints of reasonable people concerning software protection.

In reality, the only people who could object are those who would like the option of stealing your company's product.

*"...eliminating the rationale for copy-busting..."*

Since The BLOCK allows a user to make unlimited archival copies the rationale for copy-busting programs is eliminated.

The BLOCK is fully protected by federal patent law rather than the less effective copyright statutes. The law clearly prohibits the production of work-alike devices to replace The BLOCK.

The BLOCK attaches to any communications port of virtually any microcomputer. It comes with a unique customer product number programmed into the circuit.

The BLOCK is transparent to any device attached to the port. Once it is in place users are essentially unaware of its presence. The BLOCK may be daisy-chained to provide security for more than one software package.

Each software developer devises their own procedure for accessing The BLOCK to confirm a legitimate user. If it is not present, then the program can take appropriate action.

*"...possibilities... limited only by your imagination..."*

The elegance of The BLOCK lies in its simplicity. Once you understand the principle of The BLOCK, hundreds of possibilities will manifest themselves, limited only by your imagination.

Your efforts, investments and intellectual property belong to you, and you have an obligation to protect them. Let us help you safeguard what's rightfully yours. Call today for our brochure, or a demo unit."

**Software Security inc.**

870 High Ridge Road Stamford, Connecticut 06905  
203 329 8870



# PAINLESS WINDOWS.

Windows. Data Entry. Menus.  
Finally, a C programmers' tool that makes  
them as easy to use as *printf()*.  
With Greenleaf DataWindows™,  
you move in quantum leaps!

## Snazzy Window Treatments

DataWindows represents an important breakthrough in C programming tools. It sets you free so you can create exciting programs quickly and easily, saving both time and money! Developed to work with the IBM PC, XT, AT, compatibles, and MSDOS or PC DOS, DataWindows is a carefully tooled system of C functions which will jazz up your programs with unprecedented efficiency.

Greenleaf DataWindows is integrated windows, transaction data entry, pop-up, pull-down, and Lotus style menu systems with:

- **Screen Management.** You don't have to remember what's on the display or the sequence in which you put it there. DataWindows does the grunt work. There are no restrictions.
- **Transaction Data Entry.** Data entry windows can have any number of fields with sophisticated options for reading many data types. Calls are made to help, validation, and other functions. Full featured text editing, protected and mandatory fields, dBASE type picture strings, context sensitive help, validation of fields and transactions, redefinable keys, password entry, attribute control, keyboard idle and much more.
- **Device Independence.** It detects the type of display adapter your computer is using and adjusts to it automatically for CGA, EGA, or monochrome. Logical video attributes are easy to use for color or monochrome.
- **Compatibility.** Runs with Microsoft Windows and IBM TopView.
- **The Greenleaf Tradition of Quality.** Reliable products. Professional documentation that gets you up and running quickly and keeps you there. Reference card. Newsletter and Bulletin board.

IBM, Microsoft and dBase, are registered trademarks of International Business Machines, Microsoft Corporation & Ashton-Tate respectively. PC DOS, IBM PC, XT, AT, & TopView are trademarks of IBM; MSDOS and Microsoft Windows are trademarks of Microsoft Corporation.



## Stop Window Shopping

Order Today. Or call toll free for a free demo of the windows library that makes all the others obsolete.

Order any of these high performance tools by calling your dealer or 1-800-523-9830 today. Specify compiler when ordering. Add \$8 for UPS second day air, or \$5 for ground. Texas residents add sales tax. MasterCard, VISA, P.O., check, COD. In stock, shipped next day.

Greenleaf DataWindows	\$225
DataWindows Source Module	\$225
The Greenleaf Comm Library v2.0	\$185
The Greenleaf Functions v3.0	\$185
Digiboard Comm/4-II	\$325
Digiboard Comm/8-II	\$535



**GREENLEAF**

Software®

1411 LeMay Drive, Suite 101  
Carrollton, TX 75007

Call Toll Free  
**1-800-523-9830**  
In Texas and Alaska, call  
**214-446-8641**

CIRCLE 97 ON READER SERVICE CARD

## Window Dressings

■ **Simple or Complex Windows.** Up to 254 powerful overlaid windows simultaneously, all with just one kind of window to remember! Yet any window can be from one character to 32K!

■ **Easy Window Operations.** DataWindows lets you move, zoom, frame, title, change colors, titles, frames, size, location, and make windows visible or invisible at will! Functions set cursor, attributes, and write data to any window or "current window". Word wrap, auto scroll, keyboard functions.

■ **Write to Any Window Any Time.** Windows may be visible, overlaid, or invisible, and you can write to them anyway. What you write will be seen when the windows become visible.

■ **DataWindows is fast!** It writes directly to video memory (in some modes).

■ **Easy to save!** Any window, complete with attributes, can be saved on disk quickly and efficiently.

■ **Source code available. No royalties.**

## Also from Greenleaf:

### The Greenleaf Functions v3.0

The most complete, mature C language function library for the IBM PC, XT, AT and close compatibles. Includes over 225 functions — DOS, disk, video, color text and graphics, string, time/date, keyboard, disk status and Ctrl-Break functions plus many more.

### The Greenleaf Comm Library

Our 2.0 version is the hottest communications facility of its kind. Over 120 functions — ring buffered, interrupt driven asynchronous communications for up to 16 ports simultaneously with XMODEM, XON/XOFF, many many sophisticated features.

**We support all popular C compilers for MSDOS/PCDOS:** Microsoft, Lattice, Computer Innovations, Aztec, DeSmet, and others.



## COM PORT DRIVER (continued from page 46)

install excom after running the clock initialization. A bit of experimentation may be required. Excom is self-installing; just run excom to install it. If all you need is buffered input, you've finished. Excom will be properly set up by programs using the COM ports, and the existing mode command will configure excom. To tap the power of excom, you may need to access some of the extended features. If you use excom with your own programs, you can just make an *int 14h* call with *ah = 4* to set any extended options you may need. If you use excom with existing programs, you need a way to set the extended options from your keyboard or a batch file.

Listing Two, page 75, is a simple little C program called exmode that sets extended configuration parameters. Exmode works with the Microsoft 4.0 C compiler. If you have a different one, you may need to rewrite the function *int14*—it just sets registers and performs a software *int 14h*. You

can also use exmode to install and remove excom, usually useful only while testing.

Exmode is self-documenting; just run exmode and it will tell you what it can do. Note that running exmode does not add to existing extended options but sets the complete set—that is, *exmode com1 nodsr* followed by *exmode com1 nocts* is not the same as *exmode com1 nodsr nocts*. In the first case, only *nocts* is left set; in the second, both *nocts* and *nodsr* are set. COM1's settings can be cleared by running *exmode com1*. The previous examples will not change any of COM2's settings.

An interesting possibility with excom is nesting several invocations of excom. Say you have excom set up the way you want it for a shell running on your CRT and want to temporarily run a communications program. Just install excom again, set it up for communications, run the communication program, and then use exmode to remove excom. The most recent copy of excom is removed, restoring your old excom untouched.

## Availability

All the source code for articles in this issue is available on a single disk. To order, send \$14.95 to Dr. Dobb's Journal, 501 Galveston Dr., Redwood City, CA 94063 or call (415) 366-3600 ext. 216. Please specify the issue number and format (MS-DOS, Macintosh, Kaypro).

## Bibliography

Sargent, Murray; and Shoemaker, Richard L. *The IBM Personal Computer from the Inside Out*. Reading, Mass.: Addison-Wesley, 1984.  
National Semiconductor Corp. *NSC800 Microprocessor Family Data-book*. National Semiconductor, 1985.  
International Business Machines Corp. *Technical Reference—Personal Computer XT and Portable Personal Computer*. IBM, 1984.

DDJ

(Listings begin on page 64.)

Vote for your favorite feature/article.  
Circle Reader Service No. 5.

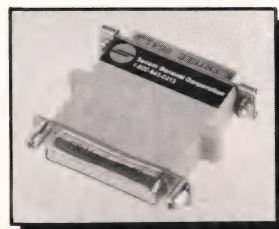
# Would you like copy protection and customer satisfaction?

Here's a better way to protect your software.  
It's called the Secom Key, and it works.

- ☐ The Key is completely transparent to the end user.
- ☐ Won't interfere with peripheral operations.
- ☐ Doesn't occupy the disk drive.
- ☐ The Key allows unlimited backup copies.
- ☐ Makes site licensing easy and auditable.
- ☐ Easily installed. Uses only 1000 bytes.
- ☐ Over 60,000 have been sold worldwide.
- ☐ Same size as RS-232 plug.
- ☐ Available in quantities for as low as \$19.95.



For more information, contact  
Secom Information Products Co.



500 Franklin Square  
1829 East Franklin Street  
Chapel Hill, NC 27707

The Secom Key...  
for real  
software  
protection.



**Secom Information Products Company**

A Subsidiary of Secom General Corporation  
Call Toll-free 1-800-843-0413

CIRCLE 394 ON READER SERVICE CARD



# Dynamic Memory Overlays for Turbo Pascal

by Steve McMahon

**B**orland International's Turbo Pascal has a significant limitation that hinders its usefulness for developing large programs. Because the compiler can generate code for only a single code segment, it imposes a 64K limit on executable code space. Turbo Pascal's overlay system, designed to circumvent this restriction, potentially allows for much larger amounts of code but at the cost of dramatically slowing program execution as overlay code is loaded from disk files.

This article describes a way to use Turbo Pascal's overlay facility in conjunction with dynamic memory to build memory overlays that can be loaded and executed far more rapidly than can disk overlays. These memory overlays are easy to set up and can optionally be left on disk at run time if sufficient memory is lacking. This technique lets you have the best of both worlds: a program that can execute in limited memory with disk overlays but that can take advantage of extra memory to load overlays into RAM.

## **Turbo Pascal's Overlay Scheme**

Borland's solution to the large-program problem is an overlay system. Programmers confronted with a program that would otherwise overrun the 64K code segment limitation can specify that the code of a group of procedures and functions should

## ***A fast and versatile way to handle large programs in Turbo Pascal***

share the same execution space in the code segment. You make this declaration by grouping the routines contiguously in the source code and preceding the declaration of each overlay routine with the Turbo reserved word *overlay*. On compilation of the program, all the routines of an overlay group are compiled to a separate overlay file, where the executing program can find them as necessary. You can specify as many overlay groups as you wish within a program. The only coding limitation is that routines within an overlay group cannot call themselves (they cannot be directly recursive) or other routines within the same overlay group. (The recursion limitation can be circumvented by calling the overlay routine from a recursive, non-overlay function or procedure.)

The advantages of this scheme are that it is easy to use and that it allows you to construct programs with code not only larger than 64K but also potentially larger than available RAM. The disadvantage is a significant performance degradation: disk activity and DOS file handling overhead are now required every time an overlay routine is executed. Careful selection and grouping of overlay routines can minimize such performance degradation, but it is still possible for a program to spend far more time reading

overlays from disk than doing anything else.

Overlay performance can be significantly enhanced by storing the overlay file on a virtual memory, or RAM, disk. Turbo Pascal's *OvrPath* procedure makes this possible by allowing run-time specification of the location of overlay files. This solution to the performance problem, though, has requirements—that a RAM disk be available, overlay files be located there, and the program be correctly informed of that location—that are unacceptable for application programs intended for use in normal DOS environments by unsophisticated operators.

A solution to the overlay performance problem that is faster, more versatile, and completely invisible to an application program's users is presented here. But first, let's take a look at how Turbo Pascal's overlay system works.

## **How Overlays Work**

When Turbo Pascal compiles a program that has overlay procedures, it reserves an overlay area in the code segment (and in the .COM file) large enough to accommodate the largest of the procedures or functions in a given overlay group. All the procedures and functions in that overlay set are then compiled for execution at the same position within the reserved overlay area. Those procedures and functions are stored all together in a separate overlay file that bears the same name as the main program but has the number of the overlay group (.000, .001, and so on) as a file-name extension.

When some other portion of the program makes a call to an over-

*Steve McMahon, P.O. Box 3262, Berkeley, CA 94703. Steve has been working with Turbo Pascal since Version 1.0. His company, SunType Publishing Systems, develops software for the newspaper industry.*



layed function or procedure, the calling code passes in registers the offset of the required overlay function or procedure inside the overlay file and the length of the code fragment to be loaded. The call is made to a fragment of code positioned at the top of the reserved overlay area. This fragment passes control to the overlay handler in Turbo Pascal's run-time package, having placed the address of the reserved overlay area on the stack. The overlay handler responds by checking to see if the code required is already in place in the overlay area (the overlay file offset of whatever code is currently in the overlay area is stored in a data area at the top of the reserved overlay space along with the name of the overlay file). If the code required is already in place, the overlay handler executes it. If not, the code is read from the overlay file: the overlay file is opened, a seek is made to the required offset, and the specified quantity of code is read into the overlay area. Finally, the file is closed, the overlay file offset of the now-current code fragment is saved in memory for future reference, and the overlay function or procedure is executed.

This overlay handling method is ripe for alteration. If the contents of the overlay file could be positioned in some otherwise unused portion of memory, then a replacement overlay handler could snatch code fragments from that area of memory rather than from a disk file. The time savings could be impressive: overlay calls would require no disk activity. Such a scheme should even provide significant speed advantages over using Borland's overlay handler in conjunction with a RAM disk (to contain the overlay file) because DOS file handling overhead would be eliminated. What a memory overlay scheme couldn't do, though, is allow for programs that are truly larger than memory because all the overlay code would need to fit in memory available to the program.

Such a scheme can be implemented with no changes to the compiler and with surprisingly little extra code. Best of all, it can be done with little modification of existing programs that use overlays. All that's necessary is to include the procedures contained in MemOvrly.Inc

(Listing One, page 78) in a program that uses overlays and to add some initialization and deinitialization code to the program. (Some limitations on the scheme are discussed later.)

### Using MemOvrly.Inc

You can include the three procedures that make up the memory overlay handler in an existing program by adding the line:

```
{ $I MEMOVRLY.INC }
```

at a point in the procedure and function declaration part of a program that uses overlays (do not nest it inside a procedure or function). Then, you must add a procedure call, probably inside the main body of the pro-

gram, to initialize the substitute overlay handler for each overlay group that you wish to use as a memory overlay group. You can add code to dispose of the dynamic memory used by the memory overlay group or groups at the end of the program. This cleanup code is particularly important if the program chains to another Turbo Pascal program because Turbo Pascal preserves the dynamic memory heap on chaining.

To set up a particular overlay group as a memory overlay group, all that's necessary is to run the *InitOverlay* procedure, passing it the address of some procedure or function in the overlay group. If, for example, an overlay group contained, among other routines, the function or procedure *One*, the entire overlay group

```
PROGRAM OverlayTest;

  (* Memory Overlay Demonstration Program. *)

  { $I MEMOVRLY.INC }

  VAR
    c : Char;

  OVERLAY PROCEDURE One;
  BEGIN
    WriteLn('This is Overlay Procedure One.');
```

END;

```
  OVERLAY PROCEDURE Two;
  BEGIN
    WriteLn('This is Overlay Procedure Two.');
```

END;

```
  BEGIN

    (Install the new overlay handler by passing it the address
    offset of ONE procedure or function from the overlay group.
    Multiple invocations for multiple overlay groups should be
    no problem.)

    InitOverlay(Ofs(One));

    REPEAT
      Write('Hit any key to run the overlays (^Z to stop): ');
      Read(Kbd, c);
      WriteLn;
      IF c <> ^Z THEN
        BEGIN
          One;
          Two;
        END;
      WriteLn;
    UNTIL c = ^Z;

    (Free up the heap space used by the replacement overlay
    handler by passing the same offset as above to the
    DisposeOverlayStorage Routine -- VITAL if you're chaining
    to another program. The heap is preserved in a chain operation.)

    DisposeOverlayStorage(Ofs(One));

  END.
```

**Example 1:** Short program demonstrating memory overlays



could be initialized as a memory overlay group with the statement:

```
InitOverlay( Ofs( One ) );
```

*Ofs* is a built-in Turbo Pascal function that returns the offset within a segment of a variable or procedure. Because you know that the reserved overlay area associated with the routine *One* is in the code segment, passing the offset of the routine is sufficient to establish where the reserved overlay area for this group is located. This instruction allocates dynamic memory sufficient to contain all the overlay code, reads that code into memory, and installs the procedure *NewOverlayHandler* as overlay handler for that group. Any overlay groups not set up in this fashion are handled by Turbo Pascal's normal overlay handler. Also, if not enough dynamic storage is available to accept the overlay code, the group is left as a normal, nonmemory overlay.

Likewise, the memory used to contain all the overlay code for a given overlay group can be reclaimed for other uses with the instruction:

```
DisposeOverlayStorage( Ofs( One ) );
```

It's crucial that you do not try to call any routine in the overlay group after issuing this instruction. The *DisposeOverlayStorage* procedure does not restore the old, nonmemory overlay handler for the specified overlay group. It merely frees up the memory previously occupied by overlay code for other uses. So, the *DisposeOverlayStorage* command is usually used only (if at all) at the end of a program. If the program does not use Turbo Pascal's *Chain* facility to chain to another Turbo Pascal program, you probably don't even need to use the *DisposeOverlayStorage* procedure. If you do need it, it should be executed once for each memory overlay group in use.

#### How It Works

When asked to initialize a memory overlay group, the *InitOverlay* procedure pulls the name of the overlay code file from the reserved overlay

area and opens the file as a Turbo Pascal untyped file. The file is sized and a check is made to make sure that enough dynamic memory is available to hold the entire file's contents with sufficient heap space left over to satisfy the program's other needs. (The amount of dynamic memory otherwise required by the program is determined by the constant *RequiredHeap*, which can be changed to reflect the needs of a particular program.)

If not enough dynamic memory is available, things go no further and the overlay group is left as a normal, nonmemory overlay group. Otherwise, heap space is allocated and the overlay file is read into memory. Then, the initialization procedure substitutes codes comprising a call instruction for the memory overlay handler for code that would normally call Turbo Pascal's native overlay handler. A data area that formerly contained the overlay file name is subsequently filled with information more pertinent to the memory overlay handling routine: the location and size of the heap space containing all the overlay code.

When a routine in the memory overlay group is needed, the call is directed to the new overlay handler. This handler uses the overlay offset and size location it receives to find the required code on the heap. The code is then moved into the reserved overlay area in the code segment with a string move and is executed.

The procedure that disposes of overlay storage on the heap is pretty straightforward. The only special thing it does is check to make sure that the overlay group it has been pointed to is actually a memory overlay group. If there wasn't adequate dynamic memory when memory overlay initialization was attempted, then the group might have been left as a normal overlay group and there would be no dynamic memory to free.

#### Limitations and Cautions

The memory overlay scheme described here does not work with overlay groups for which the overlay file is 64K or larger.

Nested overlays, created by declaring overlay groups within already overlaid functions or procedures,

cannot be made into memory overlays. This would require modification of code inside overlay files or on the heap and seems beyond the bounds of what could be reliably, simply implemented.

Turbo Pascal's *OvrPath* procedure won't work in conjunction with memory overlay procedures. This is one limitation that could be overcome without too much difficulty by anyone who understands the code in *MemOvrly.Inc*—but it's probably not necessary because the primary utility of *OvrPath* is the placement of normal overlays on RAM disks.

*MemOvrly.Inc* is highly version dependent. It definitely won't work with any version of Turbo Pascal prior to 3.0, and if Borland changes its overlay handling technique in later versions, *MemOvrly.Inc* may have to be modified to take account of the changes.

Finally, as with overlays in general, memory overlays should be used only for functions and procedures that have been thoroughly debugged. Debugging functions and procedures placed in overlay groups can be exceptionally difficult.

#### Customizing MemOvrly.Inc

You may wish to customize the memory overlay handling routines in two ways. First, the initialization routine contains no input/output error checking code.

Second, when a program needs significant amounts of dynamic memory, the *RequiredHeap* constant at the top of *MemOvrly.Inc* should be customized to reflect that fact (as should the "minimum free dynamic memory" setting on the Turbo Pascal memory usage menu).

#### Availability

All the source code for articles in this issue is available on a single disk. To order, send \$14.95 to *Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94063 or call (415) 366-3600 ext. 216. Please specify the issue number and format (MS-DOS, Macintosh, Kaypro).

DDJ

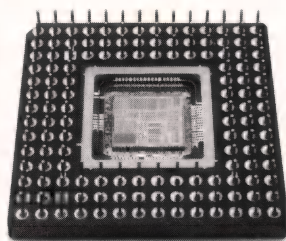
(Listing begins on page 78.)

Vote for your favorite feature/article.  
Circle Reader Service No. 6.



# A TRAINING COURSE FOR PEOPLE WHO LUST AFTER POWER.

Intel is offering three new courses on the world's most powerful 32-bit microprocessor—the 80386. Plus a new intensive course on the 80286.



The 80386

These in-depth learning sessions are designed for engineers and programmers who want to utilize the full power and potential of these lightning-fast chips.

Lectures are combined with hands-on workshops to provide real-life situations. Allowing you to apply new concepts and techniques immediately.

Courses include:

80386 System Software

80386 Programming using ASM386

High-end Microprocessor Hardware Design

The new 80286 Microprocessor Family Course

Complete training sessions and courses can be scheduled at your facility, or at our training centers. In addition to training, Intel offers hardware/software support and consultants.

For more complete course information and schedules, call toll-free (800) 548-4725.

Or to register now, contact one of the Intel Training Centers listed below.

#### Intel Training Centers

Boston Area, Westford Corp. Ctr.  
Three Carlisle Road, 1st Floor  
Westford, MA 01886  
(617) 692-1000

Chicago Area  
300 N. Martingale Road, Suite 300  
Schaumburg, IL 60194  
(312) 310-5700

San Francisco Area  
2700 San Tomas Expressway  
Santa Clara, CA 95051  
(408) 970-1700  
Washington, D.C. Area  
7833 Walker Drive, 5th Floor  
Greenbelt, MD 20770  
(301) 220-3380

**intel**<sup>®</sup>



# A Unix BBS Using Shell Scripts

by Jan L. Harrington

**T**wo years ago I assumed the responsibility of developing a BBS (now known as Scholastech Telecommunications) for Scholastech, a nonprofit organization made up of college professors and other industry professionals whose goal is to facilitate and enhance academic computing. The BBS is designed to provide a distribution mechanism for Scholastech's public-domain/shareware software collection and to further the exchange of information between educators at the college and university level.

Scholastech Telecommunications' software was originally developed and installed on an AT&T 7300 with 1-megabyte RAM and a 20-megabyte hard disk (by the time this article is published, the system will have been transferred to an AT&T 3B2 with a 72-megabyte drive). It has been in operation since October 1985, and to the best of AT&T's knowledge, it is the only BBS that has ever run on a 7300. Though the decision to use that particular machine was dictated only by circumstances (no other Scholastech hardware had the necessary configuration), the Unix operating system has provided an ideal environment in which to develop and operate a bulletin-board system. Not only is Unix multiuser and multitasking but it also contains significant support for telecommunications.

This article briefly discusses the Unix telecommunications environment and then looks in depth at the software written to implement Scho-

**Unix  
has made it  
remarkably simple  
to develop a  
bulletin-board  
system.**

lastech Telecommunications. Although it may seem like exposing the trade secret of the century, it is nonetheless true that the Unix operating system has made the development remarkably simple; the entire BBS software consists of a set of Unix V Bourne shell scripts, some supporting text files, and a public-domain XMODEM file transfer program.

## **The Unix Telecommunications Environment**

Most implementations of Unix support both electronic mail (*mail*) and intersystem file transfers (*uucp*—Unix to Unix copy). When these functions are present, the system is able to answer and place phone calls. Unix also supports *cu* (call up) operations, which turn the system issuing the call into a dumb terminal. If the system called happens to be another Unix machine, *cu* supports file transfer as well. In terms of developing a BBS, *cu* is of little use because it is unreasonable to assume that all incoming calls will be from another Unix system. *Cu*'s file transfer functions therefore cannot be used as part of BBS software.

On the other hand, the presence of facilities to manage electronic mail and *uucp* transfers means that the system can handle incoming calls from any type of system (it is nonetheless true that the full power of Unix

telecommunications can only be realized when the communication is with another Unix system). The steps for preparing a Unix system for telecommunications do not vary greatly from one implementation to another, though the details of how and where the activities occur may be different. Generally, setting up a Unix system to receive incoming calls involves the following two actions:

1. Configure communications port(s): The AT&T 7300 has two internal modems; port configurations are handled by an applications shell known as The Office (because The Office performs some undocumented actions when it configures ports, users of the 7300 are wise not to attempt to do the configuration from the Bourne shell). More commonly, however, the configuration is handled from the shell by making appropriate entries in device configuration files, though the actual names of the files will vary between implementations.
2. Verify that a *getty* process is available for the port(s): *Getty* is a Unix system process that polls devices for input; it is also the process that actually logs a user onto the system. Instructions as to how the system should respond to device activity are contained in the file *inittab*. For example, the following *inittab* is used by the 7300:

```
is:2:initdefault
rc:bootwait:/etc/rc > /dev/window
    < /dev/w1 2>&1
vid:2:respawn:/etc/getty window
    9600
:ph0:2:respawn:/etc/getty ph0 1200
:ph1:2:respawn:/etc/getty ph1 1200
000:2:respawn:/etc/getty tty000 9600
```

Jan L. Harrington, 4002 Stearns Hill Rd., Waltham, MA 02154. Jan is an assistant professor in the Computer Science Department at Bentley College. She is the author of *Macintosh Assembly Language: An Introduction*.



The first line in *inittab* puts the system in multiuser mode when the system is started; the second invokes *rc*, the shell script that handles system boot. The remaining lines are for actual devices. The colon in front of *ph0* indicates that the first modem line is inactive and should not be polled; the line in use is *ph1*. The word *respawn*, seen in the four device lines, indicates that if the *getty* process is not active when the device is polled, it should be started.

For some implementations of Unix, the same port cannot be used for both incoming and outgoing calls; the *getty* process must be inactive for outgoing lines and active for incoming lines. Configuring such systems for outgoing calls means that a colon must be placed in *inittab* at the beginning of the entry for each port that will be used to dial out, as was done above for the inactive *ph0* line. The 7300, however, kills the *getty* on a port automatically whenever an outgoing call is placed, making it possible to use the same line for both incoming and outgoing traffic.

Once one or more ports have been configured (assuming a modem is attached) and a *getty* is polling those ports, Unix answers incoming calls and takes users through the log-in process without intervention from an application program. Unix also intercepts the system log-out command, Ctrl-D, and logs users off. It is important to realize that this is not enough to configure the system for outgoing calls; outgoing calls require at least an entry in the file *L-devices* which describes the type of modem in use. *Uucp* and *mail* also require entries in the file *L.sys*, which identifies the names and phone numbers of other Unix systems that the host can call. Additional configurations may be necessary under specific Unix implementations.

What does this mean for a BBS that is to run on a Unix machine? The BBS software does not have to manage telecommunications: it does not have to poll a serial port to detect incoming calls, it does not have to log users onto the system, and it does not have to log users off the system. A simple Unix BBS needs to be concerned only with the actual functions remote users will perform after they are actu-

ally logged onto the system.

### Overview of the System

The Scholastech BBS software provides the following functions:

- file upload and download
- public message exchange
- private mail
- user help
- change of password

Each user is given a separate Unix account under what is known as the

## **The steps for preparing a Unix telecommunications system do not vary greatly.**

"restricted shell." The restricted shell prevents users from changing directories, from accessing directories not in their default *PATH* (set in a .profile, an executable shell script that is run automatically when a user logs in), and from using any commands not contained in their *rbin* directory. The *rbin* directory is also made part of the user's default path. The restricted shell therefore effectively locks BBS users into a small segment of the system, preventing them from even listing the contents of directories that are not part of the BBS.

BBS commands are each implemented as a separate shell script. There are at least four advantages to this approach:

1. The short shell scripts are easier to debug than a long, single program, regardless of whether the program is a shell script or compiled C.
2. New commands can be tested and added at any time without disturbing BBS operation.
3. Individual shell scripts simplify the logging of users' activities and the monitoring of their activities while they are on the system.
4. Shell script programming doesn't require the nearly 10 megabytes of

disk space needed by the C development system on the 7300.

There are, of course, some major disadvantages to working with separate shell scripts:

1. Each user must have his or her own account (as long as the number of users is small, this is not an enormous problem, but as the number of users rises, account management begins to consume significant amounts of time).
2. Because they are interpreted rather than compiled, shell scripts run much more slowly than would, for example, programs in compiled C.
3. The Bourne shell itself, although containing powerful flow of control statements, is weak in terms of arithmetic operations. Even simple addition must be prefaced with the command *expr* and enclosed in single quotes before the sum can be assigned to a variable.

Scholastech Telecommunications has two special accounts without passwords, new and info. The new account is designed only for on-line account requests; the info account displays information about Scholastech activities and can accept on-line sign-up for Scholastech workshops. Code for these functions is contained in their .profiles.

### BBS Directory Structure

Scholastech Telecommunications files are maintained within the directories */u/bbs* and */u/bbs/rbin* and their subdirectories. The contents of the directory in */u/bbs* are:

- *rbin* directory (contains all commands available to restricted shell users)
- MS-files directory (contains MS-DOS software)
- Unix-files directory (contains Unix software)
- Mac-files directory (contains Macintosh software)
- Uploads directory (destination for all uploaded software)
- new (log-in for on-line sign-up for new accounts)
- info (log-in for Scholastech information and on-line workshop sign-up)
- msg directory (contains support files for the public message



## UNIX BBS

(continued from page 55)

subsystem)

- **log.file** (a text file that records BBS command usage)
- **msg1** (a text file whose contents are displayed by a user's .profile)
- **directories** for each BBS user

The directory /u/bbs/rbin contains:

- **dwnld** (shell script for downloading software)
- **help** (shell script for displaying BBS instructions)
- **list** (shell script for listing the contents of a data library)
- **pmail** (shell script for sending and reading private mail)
- **readmsg** (shell script for reading public messages)
- **scan** (shell script for viewing the headers of public messages)
- **send** (shell script for sending public messages)
- **xmodem** (public-domain file transfer program; compiled C)
- **supporting text files:**

- a. **MS.list** (listing of MS-DOS data library)
- b. **Mac.list** (listing of Macintosh data library)
- c. **Unix.list** (listing of Unix data library)
- d. **help.file** (text file with BBS instructions)
- e. **user.file** (listing of system users)
- **all Unix commands** used by the BBS shell scripts (either copied or linked into this directory)

### The Shell Scripts

#### BBS User's .profile

The environment for each BBS user is configured by a short .profile:

```
PATH=/u/bbs/rbin:/u/bbs/MS-  
files:/u/bbs/Mac-files:/u/bbs/Unix-  
files:/u/bbs/Uploads  
export PATH  
PS1='> '  
cat /u/bbs/msg1  
echo
```

The .profile establishes the rbin directory and the program library di-

rectories as the user's default *PATH* (although technically there is only one restricted shell, /bin/rsh, there can be many rbin directories within a single Unix file system, each of which has a different path name and contains a different set of commands). The default prompt is also changed from the standard Bourne shell \$ to a >. Finally, the .profile displays the contents of a short welcome message (stored in msg1) and then returns control to the operating system.

Users logging in for the first time or after a long absence usually use the *help* command to display a screen full of instructions. The *help* command shell script is only two lines long:

```
echo 'who am i ! cut -f1 -d" "' 'date !  
cut -c1-c16' "help" >>/u/bbs/  
log.file  
cat /u/bbs/rbin/help.file
```

The first line writes a record to the command use log file, /u/bbs/log.file; the second simply displays the

Now with  
extended/expanded  
memory support

# PC Scheme— a simple, modern LISP for under \$100.

Texas Instruments presents PC Scheme, the \$95\* solution to your symbolic processing needs.

Whether you're an experienced LISP user, or just beginning to discover the power of symbolic programming languages, PC Scheme is the right product for you. It runs on IBM® Personal Computers, as well as the TI Professional Computer family, including the Business-Pro™ computer.

Powerful features include an optimizing incremental compiler for ease of programming and fast execution; an EMACS-like editor; extensions for debugging, graphics, and windowing; DOS-CALL capability; and a programming system for the development of object-oriented

applications—all designed to work efficiently on personal computers.

To order, or for more information, call toll-free:

**1-800-527-3500**

\*Suggested list price.  
Business-Pro is a trademark of Texas Instruments Incorporated.  
IBM is a registered trademark of International Business  
Machines Corporation.

**TEXAS  
INSTRUMENTS**

Creating useful products  
and services for you.

© 1986 TI 261765-02A



contents of a text file containing instructions.

Users who wish to change their passwords can issue the command *passwd*. This Unix command is not contained in a shell script but is made available to BBS users at their > prompt. *Passwd* prompts for the new password and then asks that it be entered again for verification. On the 7300, *passwd* also enforces format constraints—passwords must be two words separated by a special character (only passwords established by the superuser can violate those constraints).

### The Program Libraries

Three commands support user interaction with the program libraries—*list*, which displays the contents of a program library; *dwnld*, which downloads a file; and *upld*, which uploads a file.

The command *list* (Listing One, page 80) uses the Unix command *more* to display the contents of a text file that contains a listing of files available in a specific program library (lines 6, 11, and 16). Although the more widely used *cat* command also displays the contents of a text file, *more* displays the text one screen at a time (users advance by pressing the space bar or Return key) and is therefore better suited for screen displays. The Unix command to display a file directory, *ls*, could also be used to show the contents of the program libraries directly; this would eliminate the need to maintain the separate text files. In terms of security, however, *ls* is a "dangerous" command. If *ls* is present in the *rbin* directory, users who know Unix can use it from the > prompt to view the contents of that directory. They can then be aware that potentially more dangerous commands such as *mv*, which moves a file from one name or place to another, are available under the restricted shell. The best situation is to avoid the use of dangerous commands entirely, but they are essential to more than one of the BBS shell scripts.

The listings for each of the three program libraries are kept in separate files. Users must specify which program library as an argument to the command. For example, *list MS* displays the contents of */u/bbs/*

*rbin/MS.list*. The shell script traps the argument at lines 2, 7, and 12. If no argument is present (line 17), the shell script prints an error message (line 18) and exits. *List*, like all the other BBS commands, also writes a record to the command use log file (lines 4, 9, and 14).

*Dwnld* (Listing Two, page 80) transfers files from the program libraries to a remote user. The command requires two arguments—a designator for the program library and the name of the file. For example, *dwnld MS PCFILE.ARC* initiates transfer of the file */u/bbs/MS-files/PCFILE.ARC*. The first 12 lines of the *dwnld* script identify the program library. If no program library is included, the script prints an error message (line 11) and does not execute a transfer. Assuming that a valid program library has been entered, *dwnld* then verifies that a file name is present (line 13); the error message for a missing file name appears in lines 37 and 38. The presence of a file name on the command line, however, does not guarantee that the file exists. A

check for a valid file occurs on line 15 (the error message for an invalid file name appears in lines 32–34). After verifying that the requested file exists, *dwnld* writes a log record (line 17) and then prompts the user for the file transfer method (line 18). Files can be transferred as ASCII text (lines 24–29); the script simply *cats* the file to the user, who must have instructed his or her terminal emulator program to capture incoming data to disk.

Binary file transfers are performed using the XMODEM file transfer protocol (line 22). Xmodem, the program used to implement the transfers, is a stand-alone file upload and download program that can be executed from the Unix command level as well as from within a shell script. Two versions exist in the public domain, *umodem* and *uc*, both of which are available as C source code. Xmodem was obtained by downloading the *umodem* source code from an existing Unix BBS and then compiling it on the 7300 (this was done prior to loading the program libraries,

### FULL AT&T C++ for half the price of our competitors!

Guidelines announces its port of **version 1.1** of AT&T's C++ translator. As an object-oriented language, C++ includes: classes, inheritance, member functions, constructors and destructors, data hiding, and data abstraction. 'Object-oriented' means that C++ code is more readable, more reliable and more reusable. And that means faster development, easier maintenance, and the ability to handle more complex projects. C++ is **Bell Labs' answer to Ada and Modula 2**. C++ will more than pay for itself in saved development time on your next project.

# C++

## from GUIDELINES for the IBM PC: \$195

**Requires** IBM PC/XT/AT or compatible with 640K and a hard disk.

**Note:** C++ is a *translator*, and requires the use of Microsoft C 3.0 or later.

#### Here is what you get for \$195:

- The full AT&T v1.1 C++ translator.
- Libraries for stream I/O and complex math.
- "The C++ Programming Language", the definitive 327-page tutorial and description by Bjarne Stroustrup, designer of C++.
- Sample programs written in C++.
- Installation guide and documentation.
- 30 day money back guarantee.

#### To order:

send check or money order to:

**GUIDELINES SOFTWARE**  
**P.O. Box 749**  
**Orinda, CA 94563**

To order with Visa or MC,  
phone (415) 254-9393.  
(CA residents add 6% tax.)

C++ is ported to the PC by Guidelines under license from AT&T.  
Call or write for a free C++ information package.

CIRCLE 351 ON READER SERVICE CARD



when there was still enough space of the disk for the entire development system). The xmodem command line is straightforward: the argument *-sb* indicates that a file is to be sent and that its format is binary; the second argument is the path name of the file to be transmitted.

*Upld* (Listing Three, page 81) handles file transfers in precisely the opposite manner to *dwld*. ASCII uploads *echo* incoming characters to a text file (lines 14–19); to upload binary files, xmodem is invoked with the arguments *-rb*, receive binary (line 12). *Upld* also contains code to verify that a name for the file to be uploaded has been entered as an argument to the command (lines 1 and 26–27); verifies that the file name for the incoming file doesn't duplicate an existing file name in the Uploads directory (lines 1–5); and collects a description of the file (appended to the text file */u/bbs/Uploads/Doc.file*), which the *sysop* can use to quickly figure out the nature of uploaded files.

### The Public Message Subsystem

Three shell scripts support the exchange of public messages—*scan* displays the headers of all current messages, *readmsg* displays individual messages by number, and *send* sends a public message.

*Scan* is a short script that does nothing more than display the contents of a special text file, *.index*, to which an entry is added whenever a message is sent:

```
echo 'who am i : cut -f1 -d" "' 'date :
cut -c1-16' "scan" >>/u/bbs/
                                log.file
echo
echo
more /u/bbs/msg/.index
echo
```

*Scan*'s purpose is to allow users to identify the numbers of messages that might be of interest.

Messages are displayed on the screen by the command *readmsg* (Listing Four, page 81). *Readmsg* uses two simple files—*/u/bbs/.first*, which contains the number of the first message available, and */u/bbs/*

*.last*, which contains the number of the last message available—to remind users who might not have scanned the message index of valid message numbers (lines 1–9) to do so. The file names of messages stored in */u/bbs/msg* are the same as their message numbers. For example, message number 64 has the file name */u/bbs/msg/64*. Therefore, once the user enters a message number (line 12), *readmsg* can simply *cat* a file by that name (line 18). The script also traps nonexistent message numbers (lines 19–21).

The shell script *send* (Listing Five, page 82) stores messages for the public message system. The script increments the last message number (line 8) and then conducts a dialogue with the user to collect the message header (lines 11–25) and the body of the message (lines 27–34). In addition, the message header must be added to the top of the file */u/bbs/msg/.index* (that is, *scan* displays message headers in descending order by message number). To maintain the descending order, the headers are first written to a temporary file (lines 35–33). Then, the existing *.index* file is concatenated onto the end of the temporary file (line 39). Finally, the temporary file is moved on top of the old *.index*, effectively erasing the existing file (line 40). Because public messages can be addressed to groups of people (for example, *ALL* or *All 3B2 Users*), *send* does not bother to verify that the addressee is a valid system user ID.

### The Private Mail Subsystem

Private mail is built around the Unix *mail* command. The shell script *pmail* (Listing Six, page 82) builds an interface to aid users in sending and reading mail. The script first displays the four available commands (lines 4–11): *s* to send mail, *r* to read mail, *l* to see system users, and *x* to exit from *pmail*.

The *l* command is not a command normally associated with Unix mail, but sending Unix mail requires an argument of either a valid user ID or the electronic-mail name of a remote system known to the system from which the mail is sent. Unfortunately, user IDs are often unrelated to users' real names. It is therefore important for the BBS to provide some way

for users to associate system user IDs with human names. *Pmail* displays the contents of the text file */u/bbs/rbin/user.file*, which contains a listing of user names and user IDs (lines 61–63). The same information is also available from the system file */etc/passwd*. In other words, it might be possible to avoid having to maintain *user.file* by using *cut* to obtain specific fields from */etc/passwd*. The machine that supports Scholastech Telecommunications, however, also contains several accounts that are unrelated to the BBS. The code required to *grep* out the accounts that should not be displayed to BBS users far exceeds the effort to maintain *user.file*.

Sending Unix electronic mail is straightforward. The *mail* command is used with a single argument—the user ID of the recipient (although Unix mail can be directed to remote Unix systems, Scholastech Telecommunications does not make that capability available to BBS users). For example, *mail sysop* sends mail to the *sysop* account on the same system. To send private mail, the *pmail* shell script:

- displays a set of instructions that help ensure that the mail will be sent successfully (lines 19–29)
- collects the user ID of the account to which the mail is being sent (lines 31–32)
- verifies that the user ID is valid (lines 34–39)
- sends the mail by issuing the Unix *mail* command (lines 40–45)

Users working at the Unix command level can read mail by issuing the command *mail* without any arguments (there is no reason why BBS users who are knowledgeable about Unix cannot do so from their restricted shell *>* prompt). *Pmail* reads mail in exactly that way (line 55). Several options are available once a piece of mail has been displayed on the screen, including forwarding the mail to other users and saving the mail under a different file name in the user's account. *Pmail*, however, only alerts the user of the options of either deleting the message or leaving it in the user's mailbox (lines 51–54). Though knowledgeable Unix users may attempt to save the mail, BBS



# Out-Think.

But do it quietly. The business of security is extremely sensitive, so we can't really talk about what we do. And neither can the dedicated people who work here. But we will say that TRW's Command Support Division creates systems that communicate faster, more accurately, and with greater imagination. Systems that outthink the opposition and outdo our competitors. At TRW, you'll be asked to outthink...you just won't be able to think out loud.

## SENIOR COMPUTER SYSTEMS ENGINEERS

- Analyze and develop functional, performance, and security requirements for strategic command and control information systems
- Document system-level requirements and specifications
- Define a multi-computer system architecture for information management
- Structure local networks that conform with data communication standards
- Identify issues and risks associated with critical functions and new technology
- Prepare development plans, schedules and technical reviews

## SENIOR COMPUTER ENGINEERS

- Evaluate resource-sharing data communication and security capabilities of off-the-shelf computer technology
- Analyze computing and data communication workloads
- Perform computer-communication tradeoff analyses
- Identify suitable man-machine interface criteria for terminals and workstations
- Design multi-computer information management systems
- Write preliminary design specifications and interface control documents
- Design and implement prototype configurations, and experiments and exercises to evaluate them

## SENIOR HARDWARE ENGINEERS

- Apply hardware solutions for fast text-search and high-density (optical) storage
- Design and develop secure voice and telecommunication systems
- Design and implement custom computer-data communication interfaces
- Perform hardware-firmware trade-off analyses
- Define methods for built-in test and fault isolation
- Write preliminary design specifications and interface control documents

## SENIOR SOFTWARE SYSTEMS ENGINEER

- Develop functional and performance requirements for strategic command and control information systems
- Conduct functional and information flow analyses
- Document system-level requirements and specifications
- Develop software requirements for information security, communication and management
- Prepare a software development plan consistent with TRW standard procedures
- Manage all software design and development
- Prepare development plans, schedules and technical reviews

## SENIOR SOFTWARE ENGINEERS

- Analyze computing and data communication workloads
- Analyze hardware-software tradeoffs
- Design software for information security, communication and management
- Write preliminary design specifications
- Develop experiments and exercises to evaluate prototype software
- Design prototype software

## SOFTWARE ENGINEERS

- Perform software maintenance on long-term classified project in message processing, operator support, data management, system services, or facilities and operations support
- Experience on VAX or Sun with UNIX and/or ingres required

## PROGRAMMER-ANALYSTS

- Develop prototype software for information security, communication and management

## EMP HARDENING ENGINEER

- Harden electronic communication, data processing and support systems against nuclear electromagnetic pulse
- Knowledge of circuit interface protection, shielding, and grounding and isolation schemes required
- Familiarity with protection hardware, and with techniques and procedures necessary to avoid degradation of designed hardness during system deployment and operation also required

## VAX SYSTEMS ENGINEER

- Must have VAX/VMS experience, Real-Time applications, and be proficient in MACRO Assembly language

We can offer highly competitive salaries and our benefits plans are among the most attractive in the industry. Make this your chance to join a company that sees the big picture in communications. A company called TRW.

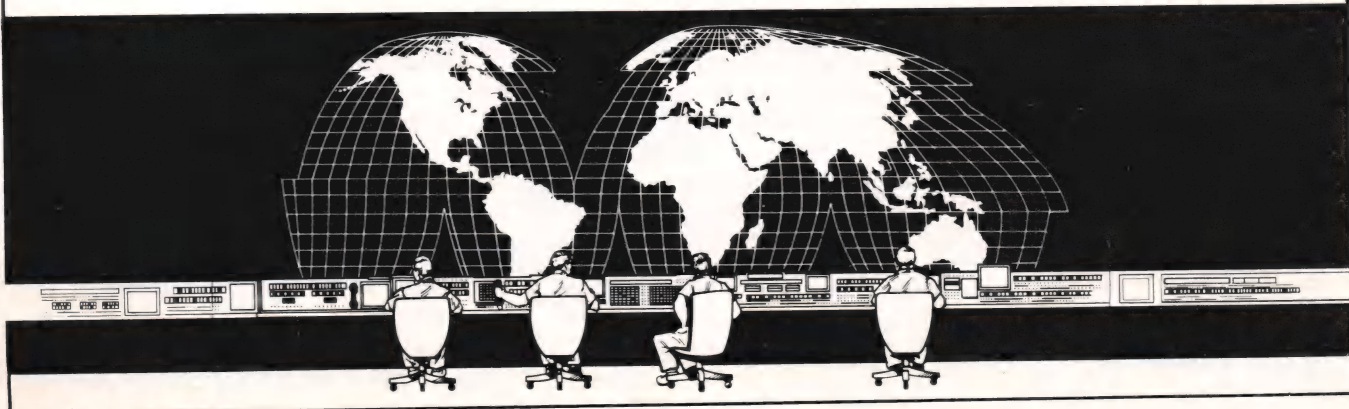
Please send resumes to: TRW Command Support Division, Dept. ST-6, One Federal Systems Park, FP2-6245, Fairfax, VA 22033. Attention: Dane Smith.

Equal Opportunity Employer. U.S. Citizenship Required.

Get the big picture at a company called TRW.

# TRW

Command Support Division  
TRW Federal Systems Group





users do not have the right to write to their own accounts (this restriction was included to conserve disk space); any attempt to save mail in a user's account will therefore fail. Knowledgeable users may, however, succeed in forwarding mail to other BBS users.

### Signing Up for a New Account

The special account `new` exists on Scholastech Telecommunications without a password. Its sole purpose is to permit prospective users to request accounts on-line. The code that manages the sign-up process is contained in the account's `.profile` (Listing Seven, page 86). Ideally, this `.profile` should automatically log users off the system once they have finished. The only way to do so that works on the 7300 (`stty 0` does not work) is to make the `.profile` the account's default shell. There is, however, an important reason why the account wasn't set up in that way; any shells other than `/bin/sh` (the standard Bourne shell) and `/bin/rsh` (the standard restricted shell) are assumed to be extremely restricted. In particular, they do not permit any redirection of output. That means that data cannot be sent to a disk file, something that is essential if user account request data are to be stored. Instead, the new account is established as a standard restricted shell account. Its default path is set to an `rbin` directory that contains only the two harmless commands `echo` and `cat` (line 1); its level 1 prompt is set to the phrase "Type CNTRL-D now ..." (line 2). When the `.profile` ends or if a user breaks out of the `.profile` with the delete key, even those who know Unix will be powerless because the commands in the `rbin` directory permit nothing but the display of text to the screen. The prompt continually reminds them of the key sequence to exit from the system.

Though somewhat lengthy for a `.profile`, the account request shell script is quite simple. It first verifies that the user wants to sign up for an account (line 5). Assuming the user does want an account, the script then enters a loop to collect name and address data (lines 13-43). After data

have been entered, the user is given a chance to view the data (lines 33-38) and to correct it if necessary (lines 42-43). This process is repeated for the user ID and initial password (lines 47-60). Note that the system's password format is not enforced during the account request process. This is because initial passwords will be established by the superuser, bypassing the format restrictions. Finally, the script writes the data to a text file (lines 63-71).

The new account's `.profile` does not actually establish new accounts.

## *I wrote two routines to keep records of system and command use.*

This must be done manually by completing the following steps:

1. Make an entry in `/etc/passwd` for the new user.
2. Enter a password for the new user.
3. Create a directory for the new user.
4. Copy the BBS `.profile` into the new user's directory.
5. Make an entry in `/u/bbs/rbin/user.file` for the new user.

Steps 1 and 2 must be performed by the superuser. The remaining steps are performed by the system operator (the account sysop). The sysop account owns all BBS accounts and their files. This prevents BBS users from viewing and/or modifying their own `.profiles`, an additional security measure often used along with the restricted shell.

### Providing Organizational Information

The shell scripts that you have seen up to this point are generic; they might be used to support just about any BBS. The `info` account, however, provides a function needed by Scho-

lastech that isn't typically found with most bulletin boards. `Info's .profile` (Listing Eight, page 88) works much like the new account's `.profile` to display information about upcoming Scholastech events and to allow users to register for workshops on-line.

Like `new's .profile`, the script first establishes a special `rbin` directory as the account's default path (line 1); `info's .profile` contains nothing but three text display commands—`cat`, `echo`, and `more`. It also changes the level 1 prompt to "Type CNTRL-D now ..." (line 2). It then prints a welcome heading (lines 4-5) and displays the contents of a text file with the welcome message (line 7). The actual information about upcoming events is stored in the more extensive `info.file`, which is displayed page by page using the Unix command `more` (line 11).

Workshop sign-ups are straightforward. Users are prompted to enter registration data (lines 20-37). The `.profile` then appends that data to a text file (lines 38-43). The registration file is later printed and turned over to the Scholastech personnel in charge of the workshops.

### System Monitoring

It is important for any BBS sysop to be aware of everything that is happening on the system. While users are logged in, their activities can be monitored with the Unix `ps` command, which displays the status of all active processes. There is, however, a further need to keep records of system and command use. Unfortunately, the 7300's implementation of Unix V does not include the standard Unix accounting functions. I therefore wrote two routines that display and reformat data from a system log file and from the log file written by each BBS shell script in order to obtain archival information.

The two shell scripts are `logs`, which displays data to the screen, and `usage`, which creates a printed report. Both rely on process beginning and ending data kept by Unix in the system file `/etc/wtmp`. Normally, `/etc/wtmp` is cleaned out every time the system is rebooted. This can, however, result in a loss of data whenever an unexpected power



## "ACTIVE PROLOG TUTOR (APT) is the best example of computer aided instruction I have seen."

Jim Loomis, Contributing Editor  
Boston Computer Society

### PROLOG CLEARLY & SIMPLY EXPLAINED

APT is designed to make learning easy, and most of all, fun. If you had the choice between reading text, looking at code, or experimenting on your own, what would you choose? Well, you don't have to choose with APT.

Prolog concepts are described in text form and then demonstrated by example using the Prolog interpreter. Use the interpreter as a practice environment to try your own ideas and get immediate feedback. A single keystroke toggles you between the tutorial text and the practice interpreter. Write a program and execute it without leaving the tutorial environment!

### BUILD APPLICATIONS INTERACTIVELY

Learn by doing. The interpreter takes you step-by-step through the development of:

- an adventure game
- an intelligent genealogical database
- an order system
- an identification expert system
- a natural language user interface

As you progress, the programs get more sophisticated, building on the information you've already learned. By the end of the tutorial you have built many working Prolog applications.

The special Prolog interpreter included with APT is designed for education and is integrated with the tutorial package. It is not appropriate for production applications.

### SYSTEM REQUIREMENTS

APT requires an IBM PC, AT or compatible hardware with 512K, and PCDOS 2.0.

### SEEING IS BELIEVING

In order to understand the power of Prolog, it is important you know *how* it executes. Traditional teaching tools are ineffective in presenting what happens as Prolog executes. APT has 3 invaluable tracing mechanisms which will literally **show** you:

- the execution pattern of Prolog, and clause order importance,
- how values are passed up and down in a recursive Prolog routine, and
- how a Prolog query is processed.

### EASY TO USE

- Manipulate the tutorial with single keystrokes.
- Move to prior pages easily.
- Instantly access any chapter from the table of contents.
- Automatically begin where you left off.
- All work is automatically saved when you leave the tutorial.
- The Prolog syntax you learn is compatible with:  
the Clocksin & Mellish standard, Arity Standard Prolog, and Arity Compiler/Interpreter

### ORDER TODAY & SAVE

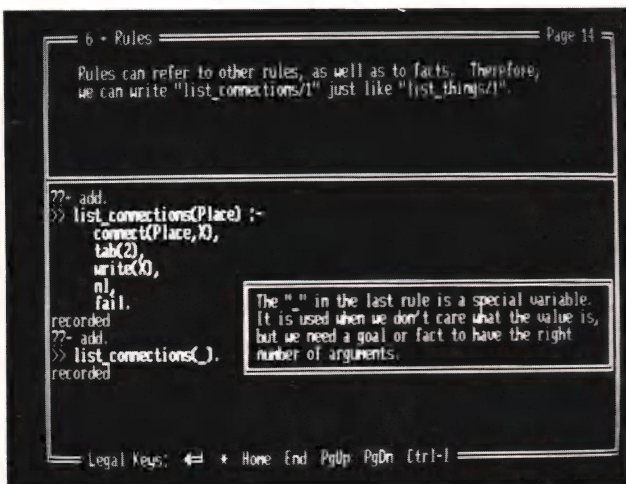
Order before June 30 and mention this ad to receive APT for \$39. That's a 40% savings from the list price of \$65.

## CALL 800-821-2492



## Solution Systems™

A typical APT screen will include a tutorial description of what is going on (top window), interpreter code (bottom window), and a pop-up window with helpful hints. Legal keys are always displayed on the bottom of the screen.





failure occurs. I therefore made a modification to the system routine *.cleanup*, which is executed by the system boot routine *rc*, and turned the line `> /etc/wtmp` into a comment to prevent its execution (I could have just as easily deleted it).

*Logs* (Listing Nine, page 90) first creates a temporary file (`/u/user.log`) that contains the user's name and the time logged on for each remote user with the Unix command *who*. A pipe to the Unix command *grep* extracts all entries that contain *ph1* (line 1). Because the device name for the internal modem being used by the BBS is *ph1*, the *grep* will include log-ins on that line and leave out all other devices (the console is usually either *w1* or *w2*, for example). If the *logs* command is issued with an argument *a* (line 2), *logs* displays the entire contents of both the temporary file (line 8) and the BBS command log file (line 15). If no argument is included, *logs* displays only the last ten lines of the two files (lines 21 and 26). Finally, *logs* removes the temporary file it created (line 29).

*Usage* (Listing Ten, page 90) creates two printed reports. The first is a columnar listing of the user's name, the date, the time on, and the time off for each log-in contained in `/etc/wtmp`. The second is simply a hard copy of the BBS command use file. *Usage* is a "dangerous" shell script; one of its functions is to reinitialize `/etc/wtmp` (line 17). It is therefore extremely important that no user other than the system operator should have any rights to the program (that is, its permission line should appear as `-rwx----`). Why shouldn't other users be allowed to read *usage*? Read rights allow another user to make off a copy of a program to put in their own account. Once the copy is made, the user becomes the owner and can execute the script, wiping out `/etc/wtmp` and destroying system use data.

The logic behind *usage* is tied to the way in which data are stored in `/etc/wtmp`. There is one record for every process that starts and every process that ends (the two exceptions are the processes *LOGIN* and *rc*). The trick to capturing log-on and log-off times is

therefore to match up the entries for any given user. Because the user running the program will not have logged off, the program must be careful to exclude that user from the report. The particular version of *usage* that appears in Listing Ten handles only remote users on the device *ph1*. In this case, the code doesn't need to worry about excluding the system operator who is running the program from the console. If, however, users on other devices are to be included, two things must happen:

***It is extremely  
important  
that no user  
besides the sysop  
has rights  
to usage.***

1. The code for assembling the report must be duplicated for each device (records for concurrent users are interleaved in `/etc/wtmp`).
2. The name of the user running the program must be excluded from the report because there will be no matching process termination record for that user.

The *usage* script first creates the system use report. It writes report headings to a temporary output file (lines 1-3) and then begins to assemble the body. The body is assembled in the following manner:

1. A log-on record for each user entering the system on *ph1* is extracted and stored in the file *temp1* (line 4).
2. The first field of the record, the user name, is *cut* from *temp1* and stored in *temp2* (line 5). This forms the leftmost column of the report.
3. The log-on time is *cut* from *temp1* and stored in *temp3* (line 6). This forms the middle column of the report.
4. A log-off record for each user entering the system on *ph1* is extracted. The log-off time is *cut* from the record and stored in the file *temp4* (line

7). This forms the rightmost column of the report.

5. The three temporary files containing the contents of the columns of the report are put together side by side with *paste* (line 8).

6. To complete the process, the report is formatted for output with the Unix *pr* command (line 9). The temporary files are removed (lines 10-14).

Records in the BBS command use log file (`/u/bbs/log.file`) are already in columnar format (they were written in such a manner by each BBS shell script). Therefore, they are simply formatted for output with *pr* (line 15). The log file is also reinitialized (line 16). Both reports are then sent to the printer (lines 18 and 19). Note that *usage* does not delete the two final output files because doing so before the files have actually been printed can disrupt the printing process.

### ***Acknowledgment***

I received both the XMODEM file transfer program and a great deal of help from David Watson, who operates a Unix/Xenix BBS in my area. Though I thanked him profusely at the time, I think he deserves a public acknowledgment for his generosity and patience in putting up with some of the elementary (dumb) questions I asked when I was first working on my BBS software.

### ***Availability***

All the source code for articles in this issue is available on a single disk. To order, send \$14.95 to Dr. Dobb's Journal, 501 Galveston Dr., Redwood City, CA 94063 or call (415) 366-3600 ext. 216. Please specify the issue number and format (MS-DOS, Macintosh, Kaypro).

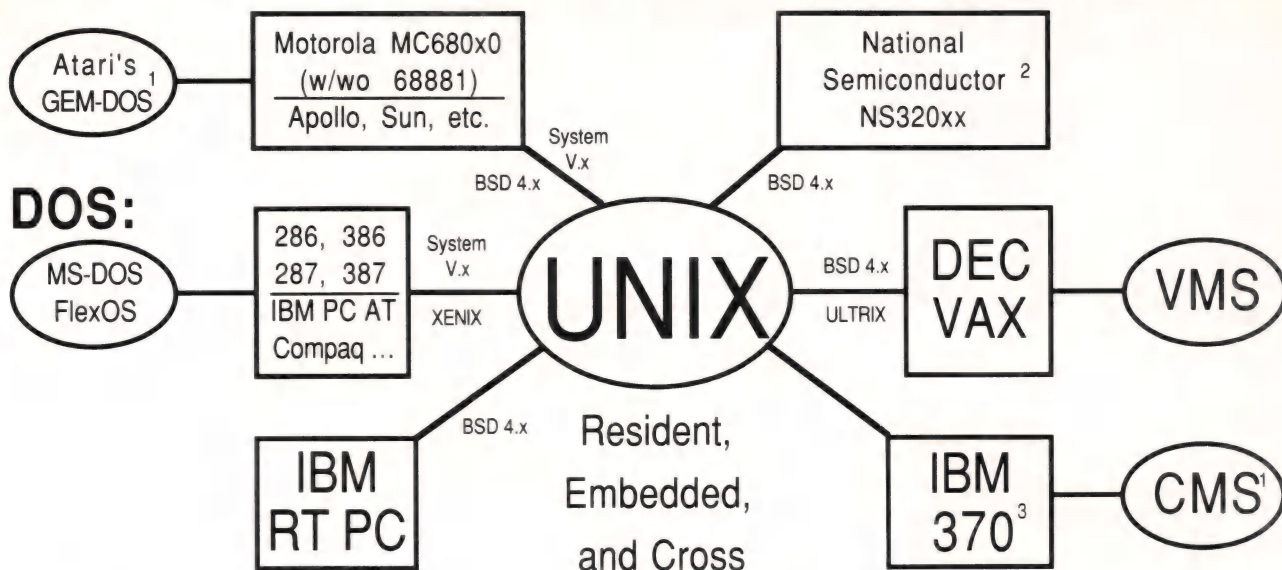
DDJ

**(Listings begin on page 80.)**

Vote for your favorite feature/article.  
Circle Reader Service No. 7.



# C and Pascal on:



We cut our teeth on UNIX, but have become famous on MS-DOS, which we enhanced with our UNIX-like **DOS Helper™** utilities: find (including tar), fgrep, cat, ls, mv, tail, uniq, and wc; and our superior optimizing compilers:

**Professional Pascal™** and **High C™** on the PC are now well-respected by organizations such as Ansa, Ashton-Tate, AutoDesk, Boeing (BCS), Daisy Systems Corp., Deloitte Haskins & Sells, Digital Research, GE, IBM, Lifetree, Migent, Multimate, NYU, Silvar-Lisco, Sky Computers, Symantec, Xerox/Ventura, ...and *Computer Language* magazine; *Dr. Dobbs' Journal*; *PC Tech Journal*; *PC Magazine*; the *Journal of Pascal, Ada, and Modula-2*...

We supply the **first, and still only**, compilers generating **32-bit protected-mode code for the 80386** under MS-DOS. And our newly upgraded MS-DOS real-mode compilers were used by Symantec for their Q&A™ product to exploit the power of the 80386 real-mode instruction set. (**Just released:** HC v1.4 and PP v2.7, May 1987.)

Our **C Validation Suite** will blow your C compiler out of the sea, while our C compiler tracks the emerging ANSI Standard and generates tighter code with far better lint-like feedback help than competing compilers.

And you'll love **Professional Pascal's** Ada-like packages, true data abstraction, C-like bit manipulation, and much more, along with the tight code that is linkable with **High C**, or other C, object modules (and vice versa).

Our **Translator Writing System (TWS)** goes far beyond LEX and YACC, with fully automatic error recovery...

All uniformly implemented on UNIX, VMS, CMS, MS-DOS, FlexOS...

**Professional software developers in need of industrial-strength tools should contact:**



MetaWare Incorporated  
903 Pacific Avenue, Suite 201  
Santa Cruz, CA 95060-4429  
(408) 429-6382 (META)  
Telex: 493-0879 (META UI)

## The Clear Choice for Large Programming Projects.

Name \_\_\_\_\_  
Company \_\_\_\_\_  
Address \_\_\_\_\_  
City, ST \_\_\_\_\_ Zip \_\_\_\_\_  
Phone: (\_\_\_\_) \_\_\_\_\_

Product:  
Platform:

Circle what interests you:

PP	HC	TWS	DOS Helper (DOS only)		
V.x	4.x	DOS	FlexOS	VMS	CMS
Sun	Apollo	Atari	VAX	370	
8086-family	80386pm	680x0	320xx		

© 1987 MetaWare Incorporated. MetaWare, High C, Professional Pascal, and DOS Helper are trademarks of MetaWare Incorporated. Others/owners: Ada/DoD; Apollo/Apollo; Atari/Atari; DEC,VAX,VMS/DEC; FlexOS,GEM-DOS/Digital Research Inc.; IBM,RT PC/IBM; MS-DOS/Microsoft; Q&A/Symantec; Sun/Sun; UNIX/AT&T.

Footnotes: 1. Atari, CMS versions available 7/87. 2. NS320xx version by special order. 3. UNIX not yet available on 370.

DDJ 06/87

CIRCLE 95 ON READER SERVICE CARD



# THE GAUSS

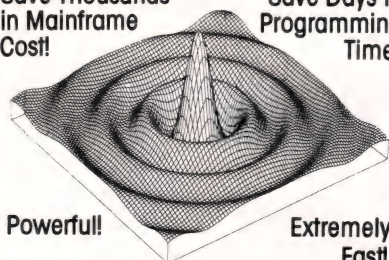
## MATHEMATICAL AND STATISTICAL SYSTEM

for IBM PC/XT/AT's and Compatibles

written by Lee E. Edlefsen and Samuel D. Jones

Save Thousands in Mainframe Cost!

Save Days in Programming Time!



Powerful!

Extremely Fast!

Easy to Learn and Easy to Use!

### The New Standard for Scientific

"I used to use FORTRAN and PASCAL for languages, TSP and Minitab for statistics, MATLAB for math, and NAG and IMSL for FORTRAN subroutines. Now I just use GAUSS."

Dr. Choon-Geol Moon  
Stanford University

• **STATISTICS** (means, frequencies, crosstabs, regression, non-parametrics, general max likelihood, non-linear least squares, simultaneous equations, logit, probit, loglinear models, & more)

• **GRAPHICS** (publication quality 2D & 3D: color, hidden line removal, zoom, pan; up to 4096 x 3120 resolution; produce Tektronix format files; output to most screen drivers, plotters, printers)

#### • PLUS:

• DATABASE MANAGEMENT

• SIMULATION • TIME SERIES/SIGNAL PROCESSING

• LINEAR PROGRAMMING

• NON-LINEAR OPTIMIZATION

• NON-LINEAR EQUATION SOLUTION

• INTERACTIVE MATRIX PROGRAMMING

• LARGE-SCALE MODULAR PROGRAMMING

• ADD YOUR OWN COMMANDS

• LINK FORTRAN, C, ASSEMBLER SUBROUTINES

Buy the **GAUSS Programming Language** by itself or a part of the **GAUSS Mathematical and Statistical System**, which includes 2D & 3D graphics plus over 200 applications programs written in the **GAUSS Program Language** for doing a variety of mathematical, statistical, and scientific tasks. Full source code is provided with these programs.

Call or Write:

**APTECH** P.O. Box 6487  
SYSTEMS, INC. Kent, WA 98064  
(206) 631-6679

#### 30-DAY MONEY BACK GUARANTEE

The GAUSS Mathematical and Statistical System..... \$350

The GAUSS Programming Language (alone)..... \$200

Shipping/handling..... \$5.00

GAUSS requires an IBM PC/AT or compatible, 320K (512K required for high resolution graphics) DOS 2.10+, and a math coprocessor.

NOT COPY PROTECTED

# COM PORT DRIVER

## Listing One (Text begins on page 42.)

```

title extended int 14 handler
name excom

;
; (C) 1987, Crystal Computer Consulting Inc.
; This software may be used freely, at your own risk,
; as long as this notice is not removed.
;
_text segment word public 'code'

assume cs:_text, ss:_text, ds:_text, es:_text

org 0100h ; com-able

start: jmp excom ; must be first

even
; misc constants
; BUFSZ equ 0100h ; buffer size
; XOFFSZ equ 00F0h ; input flow control turn off point
; XONSZ equ 00E0h ; input flow control turn on point
; DTR equ 001h ; select DTR input flow control
; RTS equ 002h ; select RTS input flow control
; XNKF equ 004h ; select XON/XOFF input flow control
; OUT2 equ 008h ; bit to set OUT2
; NOCTS equ 010h ; CTS not required to transmit
; NODSR equ 020h ; DSR not required to transmit
; B19200 equ 040h ; set baud rate to 19200
; B38400 equ 080h ; set baud rate to 38400
; C1MASK equ 010h ; com1 mask for 8259
; C2MASK equ 008h ; com2 mask for 8259

; structure for port control blocks
pcbs struc
putptr dw ? ; points to last char put
getptr dw ? ; points to next char to get
bufcnt dw ? ; number of characters in buffer
bufend dw ? ; address past buffer end
pbase dw ? ; port address
timeout db ? ; timeout outer loop
mask db ? ; mask to enable port interrupt
rxoff db ? ; set if recv has been stopped
opts db ? ; holds extended options
lchar db ? ; holds character attributes
oldier db ? ; old interrupt enable register
oldlcr db ? ; old line control register
oldmcr db ? ; old modem control register
olddll db ? ; old baud low divisor latch
olddlm db ? ; old baud high divisor latch
buf dw BUFSZ dup (?) ; the buffer
pcbs ends

; allocate storage for port control blocks
pcb pcbs 2 dup (<>) ; array of port control blocks

old0B dd ? ; old vector for int0B
old0C dd ? ; old vector for int0C
old14 dd ? ; old vector for int14
oldmsk db ? ; old mask for 8259

; baud rate table for UART initialization
bauds dw 0417h ; divisor for 110 baud
dw 0300h ; divisor for 150 baud
dw 0180h ; divisor for 300 baud
dw 00C0h ; divisor for 600 baud
dw 0060h ; divisor for 1200 baud
dw 0030h ; divisor for 2400 baud
dw 0018h ; divisor for 4800 baud
dw 000Ch ; divisor for 9600 baud
baud19 dw 0006h ; divisor for 19200 baud
baud38 dw 0003h ; divisor for 38400 baud

;
; 0Bh & 0Ch (com2 & com1) interrupt handler
; put characters into the appropriate buffer
; input flow control - stop input when buffer nears full
;
;
int0B: ; point to com2 port control block with ds:si
push ds
push si
mov si, cs
mov ds, si
lea si, pcb + size pcbs
jmp short a10

int0C: ; point to com1 port control block with ds:si
push ds
push si
mov si, cs
mov ds, si
lea si, pcb

a10: push di
push dx
push bx
push ax

; read character, if this fills buffer to XOFFSZ, then send
; XOFF or drop DTR and/or RTS as indicated by extended options

```

(continued on page 66)





# USE THE BRAINS YOUR IBM WASN'T BORN WITH.

## Right at your fingertips in CompuServe's IBM® Forums.

In the **IBM New Users Forum** you'll swap ideas with other new PC users, learn to use Forum features, and pose even basic questions to PC experts.

Our **IBM Junior Forum** gives PCjr® users a reliable source for tips on software, hardware, telecommunications, games and other interests.

In the **IBM Software Forum** you'll trade tips with other IBM PC and AT users on utility software, word processing, DOS and other operating systems.

Visit the **IBM Communications Forum** for advice on the features and compatibility of communications software and hardware, PC Bulletin Boards, micro-mainframe interfaces and more.

The **IBM Hardware Forum** addresses hardware topics of all types, plus product updates and announcements.

## Easy access to free software, including FREE uploads.

- Download first-rate, non-commercial user-supported software and utility programs.
- Upload your own programs free of connect time charges.
- Take advantage of CompuServe's inexpensive weeknight and weekend rates (when forums are most active, and standard online charges are just 10¢ per minute).
- Go online in most major metropolitan areas with a local phone call.
- And receive a **\$25.00 Introductory Usage Credit** with purchase of your CompuServe Subscription Kit.

## Information you simply can't find anywhere else.

Use the **Forum Message Board** to send and receive electronic messages, and pose specific questions to other IBM and compatible owners.

Join ongoing, real-time discussions in a **Forum Conference**.

Search our unparalleled **Forum Data Libraries** for free software, user tips, transcripts of online conferences and more.

## Enjoy other useful services like:

- **Popular Computer Magazines**—electronic editions, for your reading pleasure. Including *Dr. Dobb's Journal* and *Computer Language*.
- **Other CompuServe Forums**—supporting LOTUS® products like *Symphony™* and *1-2-3™*, Borland International®, Ashton-Tate®, Digital Research®, MicroPro®, Microsoft®, Software Publishing® and others.

## All you need is your IBM or IBM-compatible computer and a modem ...or almost any other computer.

To buy your Subscription Kit, see your nearest computer dealer. Suggested retail price is \$39.95. To receive our free brochure, or to order direct, call 800-848-8199 (in Ohio, call 614-457-0802). If you're already a CompuServe subscriber, type GO IBMNET (the IBM Users Network) at any ! prompt to see what you've been missing.

## CompuServe®

Information Services, P.O. Box 20212  
5000 Arlington Centre Blvd., Columbus, Ohio 43220  
**800-848-8199**  
In Ohio, Call 614-457-0802  
An H&R Block Company

CIRCLE 237 ON READER SERVICE CARD



## A Reconfigurable Programmer's Editor With Source Code

ME is a high quality programmer's text editor written specifically for the IBM PC. It contains features only found in the more expensive programmer's text editors. These features include:

- Multiple Windows
- Column cut and paste
- Line marking for source code
  - Regular Expressions
  - C-like Macro Language
- Reconfigurable Keyboard
- Capture your DOS session
- Run your compiler and examine errors
- Comes with useful precompiled macros

New commands and features may be added to the editor by writing programs in its macro language. The language resembles C, much easier to write macros in than a LISP based language. The macro language comes with a rich set of primitives for handling strings and changing the editor environment, plus most of the flow-of-control constructs that come in C (for, while, if, break, continue).

The source code option lets you see how text editors are written. You will also learn how to write interpreters by examining the code to the macro language compiler and interpreter.

The code is written in standard C, with several key library routines written in assembler for speed. The source code option is perfect for OEMs and VARs who want to add editing or word processing capabilities to their applications.

Price for editor and on-line documentation—\$39.95

Price for editor with complete source—\$94.95  
(Please add \$2.00 for shipping and handling)

Special offer—New York Word word processor—\$29.95  
Multi-windowing, mail merge, hyphenation, math, regular expressions, TOC and index generators

# MAGMA SYSTEMS

138-23 Hoover Ave., Jamaica, NY 11435  
(718) 793-5670

CIRCLE 313 ON READER SERVICE CARD

## COM PORT DRIVER

### Listing One (Listing continued, text begins on page 42.)

```

mov     di, [si].putptr      ; buffer put pointer
mov     dx, [si].phase
in      al, dx               ; read character and clear interrupt
mov     ah, al
cmp     [si].bufcnt, XOFFSZ
jl      a40                  ; jump if buffer below turnoff point
cmp     [si].rxoff, 0
jne     a30                  ; jump if receive already disabled
test    [si].opts, XNRF
jz      a20                  ; indicate receiver now disabled
mov     al, 'S' - '@'
out     dx, al               ; send ^S

a20:    mov     bl, [si].opts
and     bl, DTR or RTS
jz      a30
add     dx, 4
in      al, dx
not     bl
and     al, bl
out     dx, al               ; drop DTR and/or RTS
sub     dx, 4

a30:    ; if the buffer is full, set overflow status and exit
cmp     [si].bufcnt, BUFSZ
jl      a40
or      word ptr [di], 0200h
jmp     short a99

a40:    ; read the port status
add     dx, 5
in      al, dx

; get new buffer pointer, adjust for wrap around if required
inc     di
inc     di
cmp     di, [si].bufend
jne     a50
lea     di, [si].buf

a50:    ; store new buffer pointer and put status/data into the buffer
mov     [si].putptr, di
xchg    ah, al
mov     [di], ax
inc     [si].bufcnt

a99:    ; send interrupt complete command to interrupt controller
mov     al, 020h
out     020h, al
pop     ax
pop     bx
pop     dx
pop     di
pop     si
pop     ds

; =====
; 14h interrupt handler
; emulate pc bios functions plus extensions
;
; dx is used to select the port for all calls
; dx = 0 for com1, dx = 1 for com2
;
; ah = 0 initialize the port, parameters in al
; al bits 7-5 set the baud rate
; 000 = 110, 001 = 150, 010 = 300, 011 = 600
; 100 = 1200, 101 = 2400, 110 = 4800, 111 = 9600
; al bits 4-3 set the parity
; 00 = none, 01 = odd, 11 = even
; al bit 2 is number of stop bits
; 0 = 1 stop bit, 1 = 2 stop bits
; al bits 1-0 set the word length
; 00 = 5, 01 = 6, 10 = 7, 11 = 8
; returns ah & al as per comm status (ah=3) below
; example: ah = 10101110 is 2400 baud,
; odd parity, 2 stop bits, 7 bit characters
;
; ah = 1 transmit the character in al, al preserved
; returns ah as per comm status (ah=3) below
;
; ah = 2 return receive character in al
; returns ah as per comm status (ah=3) below
; only bits 1-2-3-4-7 can be set
; ah having any bits set is a receive error or timeout
;
; ah = 3 return comm port status in ah & al
; ah bits are:
; b0 = data ready b1 = overrun
; b2 = parity error b3 = framing error
; b4 = break detect b5 = xmit holding reg empty
; b6 = xmit shift reg empty b7 = timeout
; al bits are:
; b0 = delta CTS b1 = delta DSR
; b2 = trail edge ring b3 = delta recv detect
; b4 = CTS b5 = DSR
; b6 = ring indicator b7 = recv signal detect
;
; ah = 4 extended options
; returns 05A5h in ax, used to identify excom
; al contains options as follows:

```

(continued on page 70)



# ...thanks!

When we founded Programmer's Connection in 1984, we dedicated ourselves to providing high quality personal service to every customer. Since then, we've quickly grown to be the leading independent dealer in this industry.

## CELEBRATE WITH US

To celebrate our success, we're offering special sale prices on many of our products. We're also featuring some select, high-quality products on this page that represent exceptional values.

As we now enter our fourth year of business, we'd like to say "thanks!" to all of you who gave us the opportunity to serve you. And to those of you who haven't yet discovered our low prices and quality service, we extend a hearty invitation to give us a try. You'll be glad you did!

Turn the page for our latest advertised price list.

Sale prices effective through June 30, 1987.

Call or write for our FREE comprehensive price guide.

**TURBOsmith** by Visual Age is a powerful debugging environment for Turbo Pascal featuring **automatic**, multi-window readout of your source programs and variables. Trace and breakpoint with single-keystroke commands and watch the values of variables and expressions change. Supports Overlays, .COM and chain files. Synchronized Machine Code window has built-in assembler for in-line work.

List \$69 Ours \$59

**PERSONAL COBOL** by Micro Focus is a fully-integrated program development system for creating sophisticated applications on your IBM Personal Computer or Personal System/2. This versatile package comes complete with a full-screen text editor, source code generator for screens, syntax checker, automatic file handling and an interactive source-level debugging facility.

List \$149 Sale \$99

**\*PURCHASE A SUPPORTED C COMPILER** and receive 50% off any of the following libraries (when purchased on the same order):

	List	Without Language	*With Language
<b>Data Management Consultants</b>			
Zview Screen Mgmt Library	\$245	\$139	\$122.50
<b>Essential Software</b>			
C Utility Library	185	119	92.50
Essential Graphics	250	183	125.00
Essential Comm Library	185	125	92.50
with Breakout Debugger	250	189	125.00

**QUALITY.** We carry the finest selection of the best programmer's development tools specifically for IBM Personal Computers and compatibles. They are the latest versions and most come with return guarantees or evaluation periods.

**SUPPORT.** Our courteous, knowledgeable, non-commissioned salespeople are always ready to assist you. Our experienced technical consultants can answer questions about products and provide sound, unbiased advice.

**PRICE.** UPS Ground shipping is **FREE** to all U.S. customers. There are no extra charges for credit cards, CODs, purchase orders, sales tax (except Ohio) or special handling (except for export preparation). Quite simply, the prices on the next two pages are all you pay.

**INTEGRITY.** We're very proud of the trust we've earned from our customers and pledge always to be worthy of it.





**arity products**

Arity Combination Package	1095	979
Expert System Development Pkg	295	229
File Interchange Toolkit	60	44
PROLOG Compiler & Interpreter	650	569
Screen Design Toolkit	60	44
SQL Development Package	295	229
Arity PROLOG Interpreter	295	229
Arity Standard Prolog	95	77

**ai - expert systems**

1st-CLASS by Programs in Motion	495	399
AutoIntelligence by IntelligenceWare	890	739
ExpertEDGE Advanced by Human Edge	2500	CALL
ExpertEDGE Professional by Human Edge	5000	CALL
Expertech II by IntelligenceWare	475	339
EXSYS Development Software by EXSYS	395	309
EXSYS Runtime System	600	469
Insight 1 by Level Five Research	95	75
Insight 2+ by Level Five Research	485	379
Intelligence/Compiler IntelligenceWare	890	739
Logic-Line Series 1 by Thunderstone	90	85
Logic-Line Series 2 by Thunderstone	125	115
Logic-Line Series 3 by Thunderstone	150	139

**ai - lisp language**

GCLISP Golden Common LISP by Gold Hill	495	CALL
GCLISP 286 Developer by Gold Hill	1190	CALL
ICLISP by Integral Quality	300	CALL
ICLISP by Integral Quality	270	CALL
Microsoft LISP Common LISP	250	149
QNIAL Combines LISP & APL by NIAL Systems	375	339
Starter QNIAL by NIAL Systems	95	89
TransLISP from Solution Systems	95	CALL
TransLISP PLUS from Solution Systems	195	CALL

**ai - prolog language**

APT Active PROLOG Tutor from Solution Systems	65	CALL
LPA microPROLOG All Varieties	CALL	CALL
MPROLOG Language Primer LOGICWARE	50	45
MPROLOG P500 by LOGICWARE	495	395
MPROLOG P550 by LOGICWARE	220	175
Prolog-86 from Solution Systems	125	CALL
Prolog-86 Plus from Solution Systems	250	CALL
Turbo PROLOG by Borland Intl	100	63
Turbo PROLOG Toolbox by Borland Intl	100	64

**ai - smalltalk language**

Smalltalk/V by Digital	99	84
EGA/VGA Color Option	100	85
Goddies Diskette	49	45
Smalltalk/Comm	49	42

**ai - texas instruments**

PC Scheme Lisp	95	84
Personal Consultant Easy	495	435
Personal Consultant Plus	2950	2589
Personal Consultant Runtime	95	84

**apl language**

APL+PLUS/PC by STSC	595	424
APL+PLUS/PC Spreadsheet Mgr by STSC	195	139
APL+PLUS/PC Tools Vol 1 by STSC	295	199
APL+PLUS/PC Tools Vol 2 by STSC	85	58
ATLAS+GRAPHICS by STSC	CALL	CALL
Financial/Statistical Library by STSC	275	189
Pocket APL by STSC	95	69
STATGRAPHICS by STSC	795	579

**assembly language**

386 ASM/LINK Cross Asm by Phar Lap	495	389
8088 Assembler w/ Z-80 Translator by 2500 AD	100	89
Advanced Norton Utilities by Peter Norton	150	99
ASMLIB Function Library by BC Assoc	149	125
asmFREE B-free Dev System by BC Assoc	395	339
Cross Assemblers Various by 2500 AD	CALL	CALL
Microsoft Macro Assembler	150	93
Norton Utilities by Peter Norton	100	59
screenplay by Flexus	100	79
Turbo EDITASM by Speedware	99	84
Unware Cross Assemblers Various by SDS	CALL	CALL
Visible Computer: 8088 Software Masters	80	64

**basic language**

87 Software Pak by Hauppauge	180	149
EXIM Services Toolkit by EXIM	50	45
Finally by Komputerwerks	99	85
Inside Track from Micro Help	65	51
MACH 2 by Micro Help	69	55
Microsoft QuickBASIC Compiler	99	63
Peeks 'n Pokes from MicroHelp	45	37
Professional BASIC by Morgan	99	68
8087 Math Support	50	42
QuickPak by Crescent Software	69	59
Scientific Subroutine Library by Peerless	125	99
Stay-Res by MicroHelp	95	73
True Basic w/ Run-time	245	179
True Basic	150	97
Run-time Module	150	97
Various Utilities	50	41
Turbo BASIC Compiler by Borland Intl	100	64

**blaise products**

ASYNCH MANAGER Specify C or Pascal	175	119
C TOOLS PLUS	175	119
EXEC Program Changer	95	65
LIGHT TOOLS for Datatight C	100	CALL
PASCAL TOOLS	125	94
PASCAL TOOLS 2	100	CALL
PASCAL TOOLS & TOOLS 2	175	119
RUNOFF Text Formatter	60	39
TURBO ASYNCH PLUS	100	CALL
TURBO POWER TOOLS PLUS	100	CALL
VIEW MANAGER Specify C or Pascal	275	179

**borland products**

EUREKA Equation Solver	100	64
Reflex & Reflex Workshop	200	128
Reflex Data Base System	150	89
Reflex Workshop	70	45
Sidekick & Traveling Sidekick	125	85
Sidekick	85	57
Traveling Sidekick	70	45
Superkey	100	64
Turbo BASIC Compiler	100	64
Turbo C Compiler	100	64
Turbo Database Toolbox	70	41
Turbo Editor Toolbox	70	41
Turbo Gameworks Toolbox	70	41
Turbo Graphix Toolbox	70	41
Turbo Jumbo Pack Combination Package	300	219
Turbo Lighting	100	64
Turbo PASCAL Numerical Methods Toolbox	100	64
Turbo PASCAL	125	85
Turbo Tutor	100	64
Turbo PROLOG Compiler	40	24
Turbo PROLOG Toolbox	100	63
Word Wizard	100	64
Word Wizard and Turbo Lighting	70	47
	150	94

**c++**

C++ by Guidelines w/ version 1.1 kernel	195	172
PforC++ Function Library by Phoenix	395	225

**c compilers**

C86PLUS by Computer Innovations	497	CALL
Dataltight C Compiler Small Model	60	43
Dataltight Developer Kit	99	74
Dataltight Optimum-C	139	99
DeSmert C w/ Debugger & Large Case	209	184
DeSmert C w/ Debugger Only	159	138
Eco-C with Graphics by EcoSoft	125	83
Lattice C Compiler ver. 3.2 from Lattice	500	249
McWilliams Let's C Combo Pack	125	99
Let's C Compiler	75	54
csd Source Level Debugger	75	54
Microsoft C with CodeView	450	269
Turbo C Compiler by Borland Intl	100	64
Unware 68000/10/20 Cross Compiler by SDS	CALL	CALL

**c interpreters**

C-terp by Gimpel, Specify compiler	300	219
C Trainer with Book by Catalystix	122	87
Instant C by Rational Systems	500	349
Introducing C by Computer Innovations	125	CALL
Run/C by Age of Reason	120	79
Run/C Professional by Age of Reason	250	157

**c utilities**

C to dBase by Computer Innovations	150	CALL
c-tree & r-tree Combo by FairCom	650	449
c-tree ISAM File Manager	395	269
r-tree Report Generator	295	199
C Windows by Syscom	100	85
C Wings by Syscom	50	43
CI ROMPac by Computer Innovations	195	CALL
dbx dBase to C Translator by Desktop AI	350	299
with Library Source Code	550	469
Various Support Utilities	CALL	CALL
EASY SCREEN by Retail Mgmt Systems	225	199
Entelekon Products	CALL	CALL
Flash-up Windows by Software Bottling	90	78
Graphic Color version by Sci Endeavors	350	282
GRAFLIB by Sutrast	175	159
HALO Graphics by Media Cybernetics	300	205
HALO Development Pkg for Microsoft	595	359
THE HAMMER by OES Systems	195	129
PANEL Forms Management by Roundhill	295	CALL
PANEL Plus by Roundhill	495	CALL
PC Lint by Gimpel Software	139	99
PLOTHI by Sutrast	175	159
PLOTHP by Sutrast	175	159
Professional C Windows by Washburn	89	79
Scientific Subroutine Library by Peerless	175	128
screenplay for C by Flexus	CALL	CALL
Vitamin C by Creative Programming	225	158
VC Screen Forms Designer	100	79
Zview by Data Mgmt Consultants	245	139

**cobol language**

COBOLspill by Flexus	New	CALL	CALL
FPLIB for Realia COBOL by BC Associates	New	149	129
Micro Focus COBOL See Micro Focus Section			
Microsoft COBOL See Microsoft Section			
Realia COBOL with RealMENU	New	1145	899
Realia COBOL		995	783
RealCICS		995	783
RM/COBOL by Ryan-McFarland		950	CALL
RM/COBOL 85 by Ryan-McFarland		1250	CALL
screenplay for COBOL by Flexus		175	129

**debuggers & profilers**

386 DEBUG Cross Debugger by Phar Lap	195	129
Advanced Trace-86 by Morgan Computing	175	115
CI Probe by Computer Innovations	225	CALL
Codesifter Profiler by David Smith	119	85
Codesmith-86 by Visual Age	145	98
DSDB7 by Soft Advances	125	99
MiniProbe by Atron	395	CALL
Periscope I with Board by Periscope	345	289
Periscope II with NMI Breakout Switch	175	139
Periscope II-X Software only	145	105
Periscope III w/Advanced Board	995	CALL
The PROFILER with Source Code by DWB	125	89
TURBOSMITH for Turbo Pascal	69	59
The WATCHER Profiler by Stony Brook	60	51

**dos utilities**

Command Plus by ESP Software	80	69
FANSI-CONSOLE by Hersey Micro	75	62
Norton Commander by Peter Norton	75	55
Scroll & Recall by Opt-Tech Data	69	59
Taskview by Sunny Hill Software	80	55

**essential products**

See First Page for Special Offers.

C Essentials by Essential Software	100	75
C Utility Library	185	119
Essential Comm Library with Debugger	250	189
Essential Comm Library Software Only	185	125
Breakout Debugger Only Any language	125	89
Essential Graphics	250	183

**forth language**

CFORTH Native Code Compiler by LMI	300	229
Forth/83 Metacompiler Specify Target	750	599
PC/Forth by Laboratory Microsystems	150	109
PC/Forth+ by Laboratory Microsystems	250	199
Advanced Color Graphics Support	100	74
Enhanced Graphics Support	200	148
Intel 8087 Support	100	74
Interactive Symbolic Debugger	100	74
Native Code Optimizer	200	148
Software Floating Point	100	74
UR/Forth by LMI	350	279
Object Module Libraries	500	395

**fortran language**

50 MORE: FORTRAN by Peerless Engr	125	95
ACS Time Series Alpha Computer Service	495	389
Essential Graphics by Essential Software	250	183
For-Winds Alpha Computer Service	90	69
Forlib-Plus Alpha Computer Service	70	44
FORTLIB by Sutrast	125	109
FORTRAN Addendum by Impulse Engr	95	85
FORTRAN Addenda by Impulse Engr	165	138
GRAFLIB by Sutrast	175	159
HALO by Media Cybernetics	300	205
I/O PRO by MEF Environmental	149	129
Microcompables Combo Package	240	215
Gramatic	135	117
Plotmatic	135	117
Microsoft FORTRAN w/ CodeView	450	289
No Limit by MEF Environmental	129	109
Numerical Analyst by MAGUS	295	CALL
PANEL by Roundhill Computer Systems	295	CALL
PLOTHI by Sutrast	175	159
PLOTHP by Sutrast	175	159
RM/FORTRAN Ryan-McFarland	595	CALL
RTC PLUS Fortran to C by Cobalt Blue	325	CALL
Scientific Subroutine Lib by Peerless	175	128
Statistician Alpha Computer Service	295	235
Statlib GL: by PSISystems	295	239
Statlib TSF: by PSISystems	295	239
Strings & Things Alpha Computer Service	70	45

**greenleaf products**

Greenleaf Comm Library	185	125
Greenleaf Data Windows	225	157
with Source Code	450	289
Greenleaf Functions	185	125

**help utilities**

HELP/Control by MDS	125	99
On-line Help from Opt-Tech	149	99
SoftScreen/HELP by Dialectic Systems	195	149

**lattice products**

Lattice C Compiler ver 3.2 from Lattice	500	249
with Library Source Code	900	495
C Cross Reference Generator	50	37
with Source Code	200	139
C-Food Smorgasbord Function Library	150	95
with Source Code	300	179
C-Sprite Source Level Debugger	175	119
Curses Screen Manager	125	85
with Source Code	250	189
dbc Specify dbc II or dbc III	250	189
with Source Code	500	356
dbc III Plus	750	CALL
with Source Code	1500	CALL
LMK Make Facility	195	138
RPG II Combo All three items below	1100	875
RPG II Compiler No Royalties	750	625
RPG II SEU Screen Entry Utility	250	199
RPG II Sort/Merge	250	199
RPG II Screen Design Aid Utility	350	309
SecretDisk File Encryption Utility	120	88
SideTalk Resident Communications	120	88
SSP/PC Scientific Subroutine Library	350	229
Text Management Utilities	120	88
TopView Toolbasket Function Library	250	178
with Source Code	500	356

**metagraphics products**

LightWINDOWS/C for Datatight C	New	95	79
MetaWINDOWS No Royalties		185	109
MetaFONTS		80	58
MetaWINDOWS/Plus		235	184
MetaFONTS/Plus		235	184
TurboWINDOW Graphics/Windows for Turbo Pascal		80	58

**micro focus products**

Micro Focus Level II COBOL w/Animator	New	495	395
Level II COBOL		349	279
Level II Animator		195	155
Micro Focus Level II COBOL/ET for UNIX	New	CALL	CALL
Micro Focus Personal COBOL	See First Page	149	99
Micro Focus Professional COBOL		2000	1595
Micro Focus VS COBOL/XENIX	New	1495	1195



<b>Support Products:</b>			
COBOL/1Q Ad hoc Report Writer.....	New	495	395
COBOL/1Q for DOS 3.X Networks.....	New	995	795
FORMS-2.....		295	235
SOURCEWRITER.....		995	795

## microport products

System VIAT by Microport Systems.....	Sale	549	459
Runtime System (Operating System).....		199	189
Software Development System.....		249	235
Text Preparation System.....		199	189
User Upgrade 3 to Unlimited Users.....		249	235

## microsoft products

Microsoft BASIC Interpreter for XENIX.....		350	209
Microsoft C with CodeView.....		450	269
Microsoft COBOL w/ COBOL Tools.....	New Version	700	429
for XENIX.....		995	609
Microsoft FORTRAN w/ CodeView.....		450	269
for XENIX.....		695	419
Microsoft Learning DOS.....		50	36
Microsoft LISP Common LISP.....		250	149
Microsoft MACH 10 w/ Mouse & Windows.....		549	369
Microsoft MACH 10 Board only.....		399	279
Microsoft Macro Assembler.....		150	93
Microsoft Mouse Bus Version.....		175	114
Microsoft Mouse Serial Version.....		195	124
Microsoft muMath Includes muSIMP.....		300	179
Microsoft Pascal Compiler.....		300	179
for XENIX.....		695	419
Microsoft QuickBASIC Compiler.....		99	63
Microsoft Sort.....		195	125
Microsoft Windows.....		99	63
Microsoft Windows Development Kit.....		500	299

## modula-2 language

IOTools by Rhodes Associates.....		80	69
with Source Code.....		950	CALL
MODULA-2 Apprentice Pkg by LOGITECH.....		99	79
MODULA-2 Magic Pkg by LOGITECH.....		99	79
MODULA-2 ROM Pkg & Cross RT Debugger.....		299	239
MODULA-2 Window Pkg by LOGITECH.....		49	39
MODULA-2 Wizard's Pkg by LOGITECH.....		199	159
REPERTOIRE for MODULA-2 by PMI.....		89	79
Object Code Only.....		19	15

## mouse products

LOGIMOUSE BUS with PLUS Pkg by LOGITECH.....		119	98
with PLUS & PC Paintbrush.....		149	119
with PLUS & CAD Software.....		189	153
with PLUS & CAD & Paint.....		219	179
LOGIMOUSE C7 with PLUS Pkg, Specify Connector.....		119	98
with PLUS & PC Paintbrush.....		149	119
with PLUS & CAD Software.....		189	153
with PLUS & CAD & Paint.....		219	179

## other languages

CCS MUMPS Single-User by MGlobal.....		60	50
CCS MUMPS Single-User/Multi-Tasking.....		150	129
CCS MUMPS Multi-User.....		450	359
Janus/ADA C Pak by R&R Software.....		95	84
Janus/ADA D Pak by R&R Software.....		900	769
Janus/ADA ED Pak by R&R Software.....		395	CALL
Marshall Pascal by Marshall Language Systems.....		189	155
Personal REXX by Mansfield Software.....		125	99
SNDBOL4+ by Carspaw.....		95	80

## other products

Dan Bricklin's Demo Pgm Software Garden.....		75	57
Dan Bricklin's Demo Tutorial.....	New	50	45
Disk Optimizer by Soflogic Systems.....		60	55
FASTBACK by 5th Generation Systems.....		179	133
Informix All Varieties by Informix.....	CALL	CALL	CALL
Instant Replay by Nostradamus.....	New Version	CALL	CALL
MKS Toolkit with vi Editor by MKS.....		139	99
Net-Tools by BC Associates.....		149	129
OPT-Tech Sort by Opt-Tech Data Proc.....		149	99
PC/TOOLS by Custom Software.....	New	49	45
Screen Machine by MicroHelp.....	New	79	59
VIEW Term Emulator by Sci Endeavors.....		150	129

## phoenix products

Pasm86 Macro Assembler Version 2.0.....		195	108
Pdisk Hard Disk & Backup Utility.....		145	99
Plantasy Pac Phoenix Combo.....		995	619
Phinish Execution Profiler.....		395	225
Pix86plus Symbolic Debugger.....		395	225
PriorCe Comprehensive C Library.....		395	225
PriorCe++ Library for Guidelines C++.....		395	225
Plink86plus Overlay Linker.....		495	289
Pmaker Make Utility.....		125	78
Pmate Macro Text Editor.....		195	108
Pre-C Link Utility.....		295	154
Ptel Binary File Transfer Program.....		195	108

## polytron products

PolyBoost The Software Accelerator.....		80	64
PolyDesk III.....	New	99	72
PolyDesk III Archivist.....	New	50	42
PolyDesk III Cryptographer.....	New	50	42
PolyDesk III Talk.....	New	70	52
PolyLibrarian Library Manager.....		99	73
PolyLibrarian II Library Manager.....		149	109
PolyMake UNIX-like Make Facility.....		149	109
PolyShell.....		149	109
Polytron C Beautifier.....		50	42
Polytron C Library I.....		99	72

Polytron PowerCem Communications.....		139	105
PolyXREF Complete Cross Ref Utility.....		219	169
PolyXREF One language only.....		129	99
PVCS Corporate Version Control System.....		395	309
PVCS Personal.....		149	109
PVMFM Polytron Virtual Memory File Mgr.....	New	CALL	CALL

## program mgmt utilities

Compact Source Print by Aldebaran.....		55	44
Interactive EASYFLOW by Haventree.....		150	125
PrintQ by Software Directions.....		89	84
Quilt Computing Combo Package.....	New Version	250	199
QMake Program Rebuild Utility.....		99	79
SRMS Software Revision Mgmt Sys.....	New Version	185	159
Source Print by Aldebaran Labs.....		75	59
TLIB by Burton Systems Software.....		100	89
Tree Diagrammer by Aldebaran Labs.....		55	48

## raima products

dbQUERY Single-User Query Utility.....		195	129
Single-User with Source Code.....	Sale	495	329
Multi-User.....	Sale	495	329
Multi-User with Source Code.....	Sale	990	659
dbVISTA Single-User DBMS.....		195	129
Single-User with Source Code.....	Sale	495	329
Multi-User.....	Sale	495	329
Multi-User with Source Code.....	Sale	990	659

## sco products

Complete XENIX System V by SCO.....		1295	994
Development System.....		595	499
Operating System Specify XT or AT.....		595	499
Text Processing Package.....		195	144
Networks for XENIX by SCO.....		595	495
SCO Professional Lotus clone for XENIX.....		795	595

## softcraft products

Btrieve ISAM Mgr with No Royalties.....		245	184
Xtrieve Query Utility.....		245	184
Report Option for Xtrieve.....		145	99
Btrieve/N for Networks.....		595	454
Xtrieve/N.....		595	454
Report Option/N for XtrieveN.....		345	269

## text editors

Brief & dBrief Combo from Solution Systems.....		250	CALL
Brief Programmer's Text Editor.....		195	CALL
dBrief Customizes Brief for dBase III.....		95	CALL
Epsilon Emacs-like editor by Luger.....		195	147
KEDIT by Mansfield Software.....		175	98
Micro/SPF by Phaser Systems.....		175	139
PC/VI by Custom Software Systems.....		149	99
SPF/PC by Command Technology Corp.....	New Version	250	CALL
Vedit by CompuView.....		145	98
Vedit Plus by CompuView.....		195	128

## turbo pascal utilities

ALICE Interpreter by Software Channels.....		95	66
DOS/BIOS & Mouse Tools by Quinn-Curtis.....		75	67
Flash-up Windows by Software Bottling.....		90	78
MACH 2 for Turbo Pascal by Micro Help.....	New	69	55
MetraByte D/A Tools by Quinn-Curtis.....		100	89
Science & Engrg Tools by Quinn-Curtis.....		75	67
Screen Sculptor by Software Bottling.....		125	91
screenplay for Turbo Pascal by Flexus.....		100	79
Speed Screen by Software Bottling.....		35	32
System Builder by Royal American.....		100	CALL
IMPEX Query Utility.....		75	CALL
Report Builder.....		75	CALL
TDebugPLUS by TurboPower Software.....		60	49
Turbo EXTENDER by TurboPower Software.....		85	64
Turbo Professional by Sunny Hill.....		70	45
TurboHALO from IMSI.....		129	98
TurboPower Utilities by TurboPower.....		95	78
TurboRef by Graco Services.....		50	45
TURBOSmith Visual Age Debugger.....	See First Page	69	59
Universal Graphics Library by Quinn-Curtis.....	New	150	119

## wendin products

Operating System Toolbox.....	Rebate Offer	99	75
PCNX Operating system.....	Rebate Offer	99	75
PCVMS Similar to WAXMS.....	Rebate Offer	99	75
XTC Text Editor w/ Pascal source.....	Rebate Offer	99	75

## xenix/unix products

Btrieve ISAM File Mgr by SoftCraft.....		595	454
C-term by Gimpel, Specify compiler.....		498	379
c-tree ISAM Mgr by FairCom.....	Sale	395	269
dbVISTA See Raima Section.....			
dBx with Library Source by Desktop AI.....		550	489
DOSIX Console Version by Data Basics.....		399	CALL
DOSIX User Version by Data Basics.....		199	CALL
Informix All Varieties by Informix.....		CALL	CALL
Lyrix by Informix.....		595	449
Micro Focus Level II Compact COBOL.....		1000	CALL
Forms-2.....		400	CALL
Level II ANIMATOR.....		600	CALL
Microport Products See Microport Section.....			
Microsoft Products See Microsoft Section.....			
PANEL Plus by Roundhill Computer Systems.....	New	CALL	CALL
REAL-TOOLS Binary Version by PCT.....		149	89
Library Source Version.....		399	289
Complete Source Version.....		499	369
RM/CORREL by Ryan-McFarland.....		1250	CALL
RM/FORTRAN by Ryan-McFarland.....		750	CALL
SCO Products See SCO Section.....			

## LOWEST PRICES

Due to printing lead times, some of our current prices may differ from those shown here. Call for latest pricing.

## FREE SHIPPING

Orders within the USA (including Alaska & Hawaii) are shipped FREE via UPS. Express shipping is available at the shipping carrier's standard rate with no rush fees or handling charges. To avoid delays when ordering by mail, please call first to determine the exact cost of express shipping.

## CREDIT CARDS

VISA and MasterCard are accepted at no extra cost. Your card is charged when your order is shipped. Mail orders please include credit card expiration date and telephone number.

## CODs AND POs

CODs and Purchase Orders are accepted at no extra cost. POs with net 30-day terms are available to qualified US accounts only.

## FOREIGN ORDERS

Shipping charges for foreign and Canadian orders are based on the shipping carrier's standard rate. Since rates vary between carriers, please call or write for the exact cost. Foreign orders (except Canada), please include an additional \$10 for customs form preparation. All payments must be made with US funds drawn on a US bank. Please include your telephone number when ordering by mail. Due to government regulations, we cannot ship to all countries.

## VOLUME ORDERS

Volume orders may qualify for additional discounts. Call us for special pricing.

## SOUND ADVICE

Our knowledgeable technical staff can answer technical questions, assist in comparing products and send you detailed product information tailored to your needs.

## 30-DAY GUARANTEE

Most of our products (excluding books) come with a 30-day documentation evaluation period or a 30-day return guarantee. Please note that some manufacturers restrict us from offering guarantees on their products. Call for more information.

## MAIL ORDERS

Please include your telephone number on all mail orders. Be sure to specify computer, operating system and any applicable compiler or hardware interface(s). Send mail orders to:

Programmer's Connection  
136 Sunnyside Street  
Hartville, OH 44632

## CALL TOLL FREE

U S..... 800-336-1166

CANADA..... 800-225-1166

## OHIO & ALASKA

(Call Collect)..... 216-877-3781

TELEX..... 9102406879

FOREIGN..... 216-877-3781

CUSTOMER SERVICE..... 216-877-1110

Hours: Weekdays 8:30 AM to 8:00 PM EST.

Ohio customers add 6% state sales tax.  
Prices are subject to change without notice.  
Copyright Programmer's Connection, Inc., 1987.

# programmer's connection



## C & PASCAL PROGRAMMERS

Blaise Computing provides a broad range of programming tools for Pascal and C programmers, with libraries designed for serious software development. You get carefully crafted code that can be easily modified to grow with your changing needs. Our packages are shipped complete with comprehensive manuals, sample programs and source code.

### C TOOLS PLUS

\$175.00

NEW! Full spectrum of general-purpose utility functions; windows that can be stacked, removed, and accept user input; interrupt service routines for resident applications; screen handling including EGA 43-line text mode support and direct screen access; string functions; and DOS file handling.

### PASCAL TOOLS/TOOLS 2

\$175.00

Expanded string and screen handling; graphics routines; easy creation of program interfaces; memory management; general program control; and DOS file support.

### VIEW MANAGER

\$275.00

Complete screen management; paint data entry screens; screens can be managed by your application program; block mode data entry or field-by-field control. Specify C or IBM/MS-Pascal.

### ASYNCH MANAGER

\$175.00

Full featured asynchronous communications library providing interrupt driven support for the COM ports; I/O buffers up to 64K; XON/XOFF protocol; baud rates up to 9600; modem control and XMODEM file transfer. Specify C or IBM/MS-Pascal.

### Turbo POWER TOOLS PLUS

\$99.95

NEW! Expanded string support; extended screen and window management including EGA support; pop-up menus; memory management; execute any program from within Turbo Pascal; interrupt service routine support allowing you to write memory resident programs; schedulable intervention code.

### Turbo ASYNCH PLUS

\$99.95

Complete asynchronous communications library providing interrupt driven support for the COM ports; I/O buffers up to 64K; XON/XOFF protocol; and baud rates up to 9600.

### RUNOFF

\$49.95

NEW! Text formatter written especially for programmers; flexible printer control; user-defined variables; index generation; and general macro facility. Crafted in Turbo Pascal.

### EXEC

\$95.00

Program chaining executive. Chain one program from another even if the programs are in different languages. Shared data areas can be specified.

**ORDER TOLL-FREE 800-227-8087!**



**BLAISE COMPUTING INC.**

2560 Ninth Street, Suite 316 Berkeley, CA 94710 (415) 540-5441

**CIRCLE 217 ON READER SERVICE CARD**

## COM PORT DRIVER

### Listing One (Listing continued, text begins on page 42.)

```

; bit 0 enables DTR input flow control
; bit 1 enables RTS input flow control
; bit 2 enables XON/XOFF input flow control
; bit 4 don't require CTS to transmit
; bit 5 don't require DSR to transmit
; bit 6 sets baud rate to 19200
; bit 7 sets baud rate to 38400
;
; ah = 5 remove excom, restore old vectors, release memory
;
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
int14:
    sti
    push    ds
    push    si
    push    dx
    push    cx
    push    bx
; point to selected com port control block with ds:si
    mov     bx, cs
    mov     ds, bx
    lea     si, pcb
    or      dx, dx
    jz      b00
    add     si, size pcbs

b00:
; get port address in dx, return if port not installed
    mov     dx, [si].pbase
    or      dx, dx
    jz      b40

; ah has request type, jump to selected routine
    or      ah, ah
    jz      b000 ; initialize
    dec     ah
    jz      b100 ; transmit a char
    dec     ah
    jnz     b10 ; receive a char
    jmp     b200

b10:
    dec     ah
    jnz     b20
    jmp     b300 ; get status

b20:
    dec     ah
    jnz     b30
    jmp     b400 ; extended initialize

b30:
    dec     ah
    jnz     b40
    jmp     b500 ; remove excom

b40:
    pop     bx
    pop     cx
    pop     dx
    pop     si
    pop     ds
    iret

; ah = 0
; initialize the port
b000:
    push    ax

; initialize the pcb, this empties the input buffers
    lea     ax, [si].buf
    mov     [si].putptr, ax
    inc     ax
    inc     ax
    mov     [si].getptr, ax
    add     ax, BUFSZ * 2
    mov     [si].bufend, ax
    sub     ax, ax
    mov     [si].bufcnt, ax
    mov     [si].rxoff, al

; assert flow control signals and enable port interrupts
    add     dx, 4
    mov     al, DTR or RTS or OUT2
    out     dx, al

    in      al, 021h
    and     al, [si].mask
    out     021h, al ; unmask com port on int controller
    sti
    sub     dx, 3
    mov     al, 1
    out     dx, al ; enable receive int on uart

; set up baud rates and character attributes
    add     dx, 2
    pop     ax
    mov     bl, al
    and     al, 01Fh
    mov     [si].lchar, al
    and     al, [si].opts
    mov     al, B19200 or B38400
    jz      b010
    call    b440 ; init only character attributes
    jmp     short b020

b010:
    mov     cl, 4
    rol     bl, cl
    and     bx, 0Eh ; isolate baud rate selection

```

(continued on page 72)



# C BRICKLIN RUN

Data Entry • Menus • Windows • Prototyping • Database • Toolkit

# C-scape

## ■ Total Screen Control/Easy to Use

C-scape is a combination screen generator and library of screen I/O functions. Written for C programmers, C-scape brings a proven approach to the need for an easy-to-learn and use, but truly powerful and flexible screen management tool.

C-scape's kernel is your most powerful ally. Without requiring parameters you'll never use, it allows you to create tailored functions with ease and simplicity. Each key is individually definable. If you know `printf()`, you can use C-scape. C-scape's kernel provides a veritable screen design and construction toolkit to rewrite our functions or to write your own.

## ■ Most Powerful Prototyping Available

C-scape offers a unique approach to prototyping your software. You may use **Dan Bricklin's Demo Program** to create, edit, and view your screens (you can even capture existing screens from other programs), and then use C-scape's **demo2c** utility to convert each screen to code.

You can design each screen with attributes such as colors, menu selections, data entry fields (including type, validation, and field naming), masking, and text, and then automatically convert the entire screen to code.

## ■ Powerful Function Library

Use C-scape's functions for Lotus-like, pull-down, or your own menu designs, automatic scrolling, pop-up windows (number limited only by RAM), logical colors, help, time and date, yes/no, tickertape fields, secure and protected fields, and many others, to turn your demo into a fully functioning and complete program in a fraction of the time spent coding screens from scratch.

C-scape's extensive library includes just about all the data entry and display functions you'll ever need, including money functions, fully definable borders, and orthogonal field movement (get the latest list by calling for more information). And modifying our functions or writing your own is easy. C-scape adjusts automatically for CGA, EGA, monochrome, and the Hercules Graphics Card Plus in RamFont mode, and optionally writes directly to video memory, so it's flexible and fast.

## ■ Bridges to Power

C-scape includes examples of how to bridge to other powerful tools such as **c-tree** and **db\_VISTA**. You'll be integrating demos to dictionaries to file handlers and database managers in no time. You can even use C-scape to provide the screen design for AI applications, using packages that support calls to C.

## ■ Clean, Complete Documentation

C-scape's documentation is a clear example of how to write for programmers in a hurry. A short introduction uses helpful examples to explain the C-scape design. Each function is documented separately. An index makes reference easy, and a quick-reference card provides a synopsis of each function.

## ■ Source Code Included/Portable/No Royalties/No Runtime License

Providing source code at no additional cost gives you the freedom to modify existing functions without raising cost as a barrier. The source code includes all the low level routines you might need to port C-scape to an unsupported machine or compiler. Speaking of barriers, you pay *no royalties or runtime license fees*, either.

## ■ Toll Free Support and Bulletin Board

To make your job even easier, we provide technical support (toll free), and access to our 24-hour Bulletin Board.

## ■ Easy Prices/Risk-Free Terms

Try C-scape for 30 days. If you are not satisfied, return it for a refund. When you register, we send you source code. Order C-scape today, or call toll free for more information. See why some very major companies are standardizing on C-scape.

C-scape (Lattice/Microsoft/others) **Only \$199**

C-scape with Dan Bricklin's Demo Program **\$259**

**NEW** C-scape with DB Demo and db\_VISTA (RAIMA) **\$425**

Please add \$3 for first class shipping; Massachusetts orders add 5% sales tax.

**Oakland Group, Inc.**   
675 Massachusetts Avenue, Cambridge, MA 02139-3309



**CALL NOW!**  
**800-233-3733**  
**617-491-7311**



## VERSION CONTROL SYSTEM

TLIB™ keeps ALL versions of your program in ONE compact library file, even with hundreds of revisions!

- **Fastest, most powerful** version control system you can buy. *Nothing else comes close!* TLIB updates libraries faster than some text editors can load and save files.
- **LAN-compatible!** Shared libraries with PC Net, Novell, etc. Check-in/out locking for multi-programmer projects.
- Synchronized control of multiple related source files.
- **Easy to use.** Menu or DOS command line parameters.
- **Frugal with disk space.** Libraries are more compact than with most other version control systems. And TLIB uses no temporary files, so you can maintain a 300K library on one 360K diskette, with room to spare, even with TLIB itself on the same disk.
- Free copy of Landon Dyer's excellent public domain MAKE utility (.EXE, plus source code for DOS & VAX/VMS).
- **Plus:** File compare utility. Virtually unlimited source file size. Date, comments with each version. Configurable user interface, and many configurable options, like: read-only libraries; automatic tab/blank conversion; insertion of revision history comment block in the source file.

PC/MS-DOS 2.x & 3.x **Just \$99.95 + \$3 s/h** Visa/MC


### BURTON SYSTEMS SOFTWARE

P. O. Box 4156, Cary, NC 27511-4156

(919) 469-3068

CIRCLE 212 ON READER SERVICE CARD

function libraries  
disassemblers  
compilers  
text editors  
text filters  
communications support  
text formatters  
interpreters  
bulletin boards  
db routines  
compiler compilers  
window packages  
assemblers  
games  
tutorials  
math packages  
link editors  
languages  
cross compilers  
pre-processors  
function libraries  
disassemblers  
compilers  
text editors



**The C Users' Group  
Library**

A Directory  
of Public Domain  
C Source Code

Send \$10  
for Directory. Write  
or call for more details  
on over 100 volumes of  
Public Domain C Source  
Code.

The C Users' Group  
PO Box 97  
McPherson, KS 67460  
(316) 241 1065

CIRCLE 181 ON READER SERVICE CARD

**ICs** **PROMPT DELIVERY!!!**  
SAME DAY SHIPPING (USUALLY)  
QUANTITY ONE PRICES SHOWN for APRIL 19, 1987

OUTSIDE OKLAHOMA: NO SALES TAX

DYNAMIC RAM		
1Mbit	1000Kx1	100 ns \$28.50
51258	256Kx1	100 ns 6.95
4464	64Kx4	150 ns 3.50
41256	256Kx1	100 ns 4.35
41256	256Kx1	120 ns 3.50
41256	256Kx1	150 ns 3.15
4164	64Kx1	150 ns 1.30
EPROM		
27512	64Kx8	200 ns \$10.50
27C256	32Kx8	250 ns 5.15
27256	32Kx8	250 ns 4.85
27128	16Kx8	250 ns 3.75
27C64	8Kx8	150 ns 4.85
2764	8Kx8	250 ns 3.50
STATIC RAM		
43256L-12	32Kx8	120 ns \$12.95
6264LP-15	8Kx8	150 ns 2.95


640K MOTHERBOARD UPGRADE: Zenith 150, IBM PC/XT, Compaq Portable & Plus, hp Vectra

8087 5 Mhz \$11.500-  
80287-8 8 Mhz \$25000

OPEN 6 1/2 DAYS, 7:30 AM-10 PM. SHIP VIA FED-EX ON SAT.

SUNDAYS & HOLIDAYS: SHIPMENT OR DELIVERY, VIA U.S. EXPRESS MAIL

SAT DELIVERY INCLUDED ON FED-EX ORDERS RECEIVED BY: Th: Std Air \$6/4 lbs Fr: P-One \$13/2 lbs

MasterCard/VISA or UPS CASH COD  
**Factory New, Prime Parts**   
MICROPROCESSORS UNLIMITED, INC.  
24,000 S. Peoria Ave., (918) 267-4961  
BEGGS, OK, 74421  
No minimum order. Please note that prices are subject to change. Shipping & insurance extra, & up to \$1 for packing materials. Orders received by 3 PM CST can usually be delivered the next morning, via Federal Express Standard Air in \$4.00, or guaranteed next day Priority One in \$13.00! All parts guaranteed.

CIRCLE 105 ON READER SERVICE CARD

## COM PORT DRIVER

### Listing One (Listing continued, text begins on page 42.)

```

b020:  call    b430                ; init baud and character attributes
      sub     dx, 3
      jmp     b300            ; return status

;
; ah = 1
; transmit the character in al

b100:  push    ax
      add     dx, 6
      mov     bh, [si].opts
      not     bh
      and     bh, MODSR or NOCTS
      ; optionally wait for DSR and/or CTS
      call    b130
      jnz     b110
      dec     dx
      mov     bh, 020h
      ; wait for the transmitter to be ready
      call    b130
      jnz     b110
      sub     dx, 5
      pop     ax
      push    ax
      ; actually send the character
      out     dx, al
      call    b310
      jmp     short b120

b110:  ; restore al and indicate a timeout (bit 7 of status, ah)
      call    b310
      or      ah, 080h

b120:  pop     cx
      mov     al, cl
      jmp     b40

;
; wait for selected status, bh is status, dx is port address

b130:  mov     bl, [si].timeout    ; delay counter

b140:  sub     cx, cx

b150:  in      al, dx
      and     al, bh
      cmp     al, bh
      jz      b160                ; status match
      loop    b150
      dec     bl
      jnz     b140
      or      bh, bh                ; timeout, set non-zero

b160:  ret

;
; ah = 2
; receive a character, put it in al

b200:  cmp     [si].bufcnt, 0
      jne     b230                ; chars in buffer, return one

; wait for a character to appear in the buffer, or return timeout
      mov     al, [si].timeout
      add     al, al                ; shorter timeout loop, so loop more

b210:  sub     cx, cx

b220:  cmp     [si].bufcnt, 0
      jne     b230
      loop    b220
      dec     al
      jnz     b210
      mov     ax, 08000h            ; timeout return value
      jmp     b40

b230:  ; if receiver is disabled and room now exists, enable receiver
      cmp     [si].rxoff, 0
      je      b250                ; receive is already enabled
      cmp     [si].bufcnt, XONSZ
      jg      b250                ; buffer fuller that turn on point
      test    [si].opts, XNXF
      jz      b240
      mov     [si].rxoff, 0        ; indicate receiver enabled
      mov     al, 'Q' - 'a'
      out     dx, al                ; send ^Q

b240:  ; it's easier to just assert DTR & RTS rather than see if needed
      add     dx, 4
      in      al, dx
      or      al, DTR or RTS
      out     dx, al                ; assert DTR and RTS

b250:  ; get the char from the buffer and update buffer get pointer
      call    b310
      and     ah, 01Eh            ; status reported with a receive
      inc     bx
      inc     bx
      cmp     bx, [si].bufend
      jne     b260
      cmp     bx, b260
      lea     bx, [si].buf
  
```



# DDJ 1986 Back Issues

## March 1986 #113 Volume XI, Issue 3

Parallel Processing—Concurrency and Turbo Pascal—What Makes DOS Fast—Minimizing Arbitrary Functions—MC68000 vs. NS32000.

## Aug. 1986 #118 Volume XI, Issue 8

Special C Issue—Benchmarking C Compilers—The Joy of Conciseness—Nearly Perfect Trees—Generics in Ada—Real-World Data Types.

## Sept. 1986 #119 Volume XI, Issue 9

Smooth Algorithms—MS-DOS Directory Traversal—Turbo Boards Review—Radix Sort—Does Turbo Prolog Measure Up—Crawling Memory Test.

## Oct. 1986 #120 Volume XI, Issue 10

80386 Programming—MS-DOS File Browsing—Converting to the 320xx—Modula-2 Compiler Review—Factoring in Forth.

## Nov. 1986 #121 Volume XI, Issue 11

Graphics Routines—The New Graphics Chips—Programming Tips in C, Modula-2, Pascal, and Ada—68k Graphics.

## Dec. 1986 #122 Volume XI, Issue 12

Multitasking—32000 Assembler—Comparing String Comparisons—Turbo Pascal Procedural Parameters.

## Jan. 1987 #123 Volume XII, Issue 1

Annual 68K Issue, 68K Mini Forth, OS-9 Operating System, Mac and Amiga Interface Programming.

## Feb. 1987 #124 Volume XII, Issue 2

Editors and Assemblers.

## Mar. 1987 #125 Volume XII, Issue 3

Data Compression Techniques

## April 1987 #126 Volume XII, Issue 4

Artificial Neural Networks

Other issues are also available. Please inquire.

TO ORDER: Return this coupon with your payment to: M&T Books, 501 Galveston Dr. Redwood City, CA 94063. Or, call TOLL-FREE 800-533-4372 Mon-Fri 8a.m.-5p.m. In CA call 800-356-2002

Please send the issues circled:

113 118 119 120 121  
122 123 124 125 126

Price: 1 issue—\$5.00. 2–5 issues—\$4.50 each. 6 or more—\$4.00 each. (There is a \$10.00 minimum for charge orders.)

Subtotal \_\_\_\_\_

CA residents add sales tax \_\_\_\_\_ % \_\_\_\_\_

Outside U.S., add \$ .50 per issue \_\_\_\_\_

TOTAL \_\_\_\_\_

Name \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_

State \_\_\_\_\_ Zip \_\_\_\_\_

☐ Check Enclosed. Make Payable to M&T Publishing.  
Please Charge my ☐ VISA ☐ M/C ☐ AMEX

Card No. \_\_\_\_\_

Exp. Date \_\_\_\_\_

Signature \_\_\_\_\_

3128E

```

mov     [si].getptr, bx      ; updated pointer
dec     [si].bufcnt
jmp     b40

;
; ah = 3
; get port status
b300:   call    b310
        inc     dx
        in      al, dx      ; modem status
        jmp     b40

;
; get line status from input buffer and/or hardware into ah
b310:   mov     dx, [si].pbase
        add     dx, 5
        in      al, dx      ; line status
        mov     ah, al
        cmp     [si].bufcnt, 0
        je      b320

; when chars in buffer, fix status as per status in buffer
; return as above plus bx = getptr, al = character
        mov     cl, al
        mov     bx, [si].getptr
        mov     ax, [bx]      ; status: data (ah:al)
        mov     ch, 01Eh      ; these status bits from buffer
        and     ah, ch
        not     ch
        and     cl, ch
        or      ah, cl
        or      ah, 1        ; char in buffer, show data ready
b320:   ret

;
; ah = 4
; extended initialization
b400:   ; save the options in the pcb
        mov     [si].opts, al

; if an extended baud rate, set up the uart
        add     dx, 3
        test    al, B19200
        jz      b410
        mov     bx, baud19 - bauds
        call    b430
        jmp     short b420

b410:   test     al, B38400
        jz      b420
        mov     bx, baud38 - bauds
        call    b430

b420:   mov     ax, 05A5Ah      ; magic value to identify excom
        jmp     b40

;
; init the baud rate and character attributes
b430:   mov     al, 080h
        out     dx, al
        mov     ax, [bx+bauds]
        sub     dx, 3
        out     dx, al      ; set low order divisor on uart
        inc     dx
        mov     al, ah
        out     dx, al      ; set high order divisor on uart
        inc     dx
        inc     dx

b440:   mov     al, [si].lchar
        out     dx, al      ; set character attributes
        ret

;
; ah = 5
; remove excom
b500:   push     ds
        push     ds

; restore status of interrupt controller and uarts
        cli
        mov     ah, oldmask
        and     ah, C1MASK or C2MASK
        in      al, 021h
        and     al, not (C1MASK or C2MASK)
        or      al, ah
        out     021h, al
        sti
        lea     si, pcb      ; com1 pcb
        call    b510        ; restore com1 uart status
        lea     si, pcb + size pcbs ; com2 pcb
        call    b510        ; restore com1 uart status
        jmp     short b530

; subroutine to restore uart status
b510:   mov     dx, [si].pbase
        or      dx, dx
        jz      b520        ; no port

```

(continued on next page)



**"What's the longest time in the world?"**

"Waiting to finish a file transfer."

**"What's the answer?"**

"RamNet. This sophisticated software handles all your data transfers, E-Mail, BBS, and other communications functions—All automatically in the background—while you continue to run any program, do any other task in the foreground—All without interruption—All for \$149."

**"How do I find out more?"**

"Call the RamNet BBS at (212) 889-6438."

**RamNet**

Software Concepts Design  
594 Third Ave. New York, NY 10016  
(212) 889-6431

Major Credit Cards Accepted

CIRCLE 393 ON READER SERVICE CARD

# COM PORT DRIVER

**Listing One** (Listing continued, text begins on page 42.)

```

add     dx, 3
mov     al, 080h
out     dx, al                      ; set access to baud divisors
sub     dx, 3
mov     al, [si].olddll
out     dx, al                      ; baud divisor low
inc     dx
mov     al, [si].olddlm
out     dx, al                      ; baud divisor high
add     dx, 2
mov     al, [si].oldlcr
out     dx, al                      ; line control register
inc     dx
mov     al, [si].oldmcr
out     dx, al                      ; modem control register
sub     dx, 3
mov     al, [si].oldier
out     dx, al                      ; interrupt enable register
b520:   ret
b530:   ; restore old int 0Bh vector
        lds     dx, old0B
        mov     ah, 025h
        mov     al, 0Bh
        int     021h

        ; restore old int 0Ch vector
        pop     ds
        lds     dx, old0C
        mov     ah, 025h
        mov     al, 0Ch
        int     021h

        ; restore old int 14h vector
        pop     ds
        lds     dx, old14
        mov     ah, 025h
        mov     al, 14h
        int     021h

        ; free excom's memory, this memory
        push    es
        mov     ax, cs
        mov     es, ax
        mov     ah, 049h
        int     021h
        pop     es
        jmp     b40

; =====
;
;         installation entry point, must be at file end
;         initialize and stay resident
;
; =====
excom:

        ; initialize constant parts of each pcb
        mov     ax, 040h
        mov     es, ax                      ; bios data segment
        sub     di, di                      ; offset to com1 base address
        mov     bx, 07Ch                    ; offset to com1 timeout
        mov     cl, not C1MASK              ; com1 interrupt mask
        lea     si, pcb                     ; com1 pcb address
        call    c00                         ; init com1
        inc     di
        inc     di
        inc     bx
        mov     cl, not C2MASK              ; offset to com2 base address
        add     si, size pcbs               ; offset to com2 timeout
        call    c00                         ; com2 interrupt mask
        call    c20                         ; com2 pcb address
        jmp     short c10                   ; init com2

        ; subroutine to initialize parts of a pcb
c00:
        mov     dx, es:[di]                 ; port base from bios
        mov     [si].pbase, dx
        mov     al, es:[bx]
        mov     [si].timeout, al           ; copy timeout from bios
        mov     [si].mask, cl              ; save interrupt mask
        ret

c10:
        ; save old status of interrupt controller and uarts
        in     al, 021h
        mov     oldmask, al
        lea     si, pcb                     ; com1 pcb
        call    c20                         ; save old com1 uart status
        lea     si, pcb + size pcbs         ; com2 pcb
        call    c20                         ; save old com1 uart status
        jmp     short c40

        ; subroutine to save uart status
c20:
        mov     dx, [si].pbase
        or      dx, dx                      ; no port
        jz      c30
        add     dx, 3
        in     al, dx
        and     al, 07Fh
        mov     [si].oldlcr, al             ; line control register
        mov     al, 080h
        out     dx, al                      ; set access to baud divisors
        sub     dx, 3

```



```

in      al, dx
mov     [si].olddll, al      ; baud divisor low
inc     dx
in      al, dx
mov     [si].olddlm, al      ; baud divisor high
add     dx, 2
mov     al, [si].oldlcr
out     dx, al               ; restore register access
inc     dx
in      al, dx
mov     [si].oldmcr, al      ; modem control register
sub     dx, 3
in      al, dx
mov     [si].oldier, al      ; interrupt enable register

c30:    ret

c40:    ; release environment memory, not used
mov     bx, 02Ch
mov     ax, [bx]
mov     es, ax               ; address of env
mov     ah, 049h
int     021h                ; free memory

; save old int 0Bh vector, install new handler
push    ds
mov     ax, cs
mov     ds, ax
mov     ah, 035h
mov     al, 0Bh
int     021h
mov     word ptr old0B, bx
mov     ax, es
mov     word ptr old0B + 2, ax
lea     dx, int0B
mov     ah, 025h
mov     al, 0Bh
int     021h

; save old int 0Ch vector, install new handler
mov     ah, 035h
mov     al, 0Ch
int     021h
mov     word ptr old0C, bx
mov     ax, es
mov     word ptr old0C + 2, ax
lea     dx, int0C
mov     ah, 025h
mov     al, 0Ch
int     021h

; save old int 14h vector, install new handler
mov     ah, 035h
mov     al, 014h
int     021h
mov     word ptr old14, bx
mov     ax, es
mov     word ptr old14 + 2, ax
lea     dx, int14
mov     ah, 025h
mov     al, 014h
int     021h
pop     ds

; exit and keep everything above the entry point 'excom'
lea     dx, excom
mov     cl, 4
shr     dx, cl
inc     dx
mov     ah, 031h
mov     al, 0
int     021h                ; terminate-and-stay-resident

_text   ends
end      start
endm

```

End Listing One

## Listing Two

```

#include <stdio.h>
#include <dos.h>

/*
(C) 1987, Crystal Computer Consulting Inc.
This software may be used freely, at your own risk,
as long as this notice is not removed.
*/

#define EXINIT      4          /* ah value for extended init */
#define EXCOM       0x5A5A     /* magic value to identify excom */
#define DTR         0x01       /* DTR input flow control */
#define RTS         0x02       /* RTS input flow control */
#define XONXOFF     0x04       /* XON/XOFF input flow control */
#define CTS         0x10       /* CTS not required for transmit */
#define DSR         0x20       /* DSR not required for transmit */
#define B19200      0x40       /* set baud rate to 19200 */
#define B38400      0x80       /* set baud rate to 38400 */
#define COMINIT     0x100      /* port was selected */

typedef struct {
    char      *str;             /* command string */
    void      (*fnct)();        /* command to execute */
    int       arg;              /* argument to pass */
} CMDS;

char
inmsg = "excom installed",
remmsg = "excom not installed",
helpmsg;

```

(continued on next page)

# Dr. Dobb's Toolbook of 68000 Programming

This complete collection of practical programming tips and techniques for the 68000 family includes the best articles on 68000 programming ever published in Dr. Dobb's, along with much new material. You'll learn about the most important features of the 68000 microprocessor from a full description of its history and design. And, useful applications and examples will show you why computers using the 68000 family are easy to design, produce and upgrade. Contents include:

an introduction to the 68000 family

- 68000 Instruction Set

Development Tools

- Bringing Up to the 68000: A First Step
- A 68000 Cross-Assembler
  - Useful 68000 Routines and Techniques
- A Simple Multitasking Kernel for Real-Time Applications
- The Worm Memory Test
- A Mandelbrot Program for the Macintosh

All programs are also available on disk!

Specify MS-DOS, CP/M 8", Osborne, Macintosh, Amiga, Atari 520st.

## 68000 Cross Assembler

An executable version of the 68000 Cross-Assembler discussed in the book is also available, complete with source code and documentation. Requires CP/M 2.2 with 64k or MS-DOS with 128k. Specify 8" SS/SD, Osborne, MS-DOS.

YES! Please send me the **68000 Toolbook** for \$29.95 \_\_\_\_\_

Send me the **68000 Toolbook** & disk for \$49.95 \_\_\_\_\_

Send me the 68000 Cross Assembler for \$25.00 \_\_\_\_\_

Subtotal \_\_\_\_\_

CA residents add sales tax \_\_\_\_\_ % \_\_\_\_\_

Add \$2.25 per item for shipping \_\_\_\_\_

TOTAL \_\_\_\_\_

Specify disk format. See above. \_\_\_\_\_

Name \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_

State \_\_\_\_\_ Zip \_\_\_\_\_

☐ Check enclosed. Make payable to M&T Publishing. Please Charge my ☐ VISA ☐ M/C ☐ AMEX

Card No. \_\_\_\_\_

Exp. Date \_\_\_\_\_

Signature \_\_\_\_\_

3128F





**SQL Compatible Query System** adaptable to any operating environment.

**CQL Query System.** A subset of the Structured English Query Language (SEQUEL, or SQL) developed by IBM. Linked files, stored views, and nested queries result in a complete query capability. File system interaction isolated in an interface module. Extensive documentation guides user development of interfaces to other record oriented file handlers.

#### Portable Application Support System

**Portable Windowing System.** Hardware independent windowing system with borders, attributes, horizontal and vertical scrolling. User can construct interface file for any hardware. Interfaces provided for PC/XT/AT (screen memory interface and BIOS only interface), MS-DOS generic (using ANSI.SYS), Xenix (both with and without using the curses interface), and C-library (no attributes).

**Screen I/O, Report, and Form Generation Systems.** Field level interface between application programs, the Query System, and the file system. Complete input/output formatting and control, automatic scrolling on screens and automatic pagination on forms, process intervention points. Seven field types: 8-bit unsigned binary, 16 bit signed binary, 16 bit unsigned binary, 32 bit signed binary, monetary (based on 32 bit binary), string, and date.

Including Source Code

**\$395.00**

File System interfaces include  
C-tree and BTRIEVE.

HARDWARE AND FILE SYSTEM  
INDEPENDENT

**KURTZBERG  
COMPUTER SYSTEMS**

41-19 BELL BLVD.  
BAYSIDE, N.Y. 11361

VISA/Master Charge accepted  
**(718) 229-4540**

\*C-tree is a trademark of FairCom

IBM, SEQUEL, PC, XT, AT are trademarks of IBM Corp.  
MS-DOS and Xenix are trademarks of Microsoft Corp.  
CQL and the CQL Logo are trademarks of Kurtzberg Computer Systems.

## COM PORT DRIVER

### Listing Two (Listing continued, text begins on page 42.)

```
int      init1 = 0,          /* extended init for com1 */
         init2 = 0,          /* extended init for com2 */
         portnum = -1;      /* port to initialize */

void      exit(), install(), remove(), exinit(), setport();

CMDS      cmds[] = {
    "install", install, 0,
    "remove", remove, 0,
    "com1", setport, 0,
    "com2", setport, 1,
    "dtr", exinit, DTR,
    "rts", exinit, RTS,
    "xnxfn", exinit, XNXFIN,
    "nocts", exinit, NOCTS,
    "nodsr", exinit, NODSR,
    "19200", exinit, B19200,
    "38400", exinit, B38400,
    NULL
};

main(argc, argv)
int      argc;
register char **argv;
{
    register CMDS *cmdp;

    /* is excom installed */
    if (int14(EXINIT, 0, 0) == EXCOM && int14(EXINIT, 0, 1) == EXCOM)
        helpmsg = inmsg;
    else
        helpmsg = remmsg;

    if (argc == 1)
        help(helpmsg);

    while (**argv != NULL) {
        /* look up the argument in the command table */
        for (cmdp = cmds; cmdp->str != NULL; ++cmdp)
            if (strcmp(*argv, cmdp->str) == 0)
                break;

        if (cmdp->str == NULL)
            help("bad command");

        else {
            /* excom must be installed to set options */
            if (helpmsg == remmsg && cmdp->fnct != install)
                help(helpmsg);

            /* execute the command */
            (*cmdp->fnct)(cmdp->arg);
        }

        if ((init1 & (B19200 | B38400)) == (B19200 | B38400)
            || (init2 & (B19200 | B38400)) == (B19200 | B38400))
            help("ambiguous baud rate setting");

        /* actually send the init bits to excom */
        if ((init1 & COMINIT) != 0)
            int14(EXINIT, init1, 0);

        if ((init2 & COMINIT) != 0)
            int14(EXINIT, init2, 1);

        exit(0);
    }

    /* install excom */
    void
    install()
    {
        system("excom");
        helpmsg = inmsg;
    }

    /* remove excom */
    void
    remove()
    {
        int14(5, 0, 0);
        helpmsg = remmsg;
    }

    /* collect up the init bits for each port */
    void
    exinit(thebit)
    int
    {
        thebit;

        if (portnum == -1)
            help("no port selected");

        else if (portnum == 0)
            init1 |= thebit;

        else
    }
```



```

setport(newport)
int
(
    Newport;
    portnum = newport;
    if (portnum == 0)
        init1 |= COMINIT;
    else
        init2 |= COMINIT;
)

/* perform int 14h with ah, al & dx as passed, return ax value */
int14(cmd, val, port)
int
(
    cmd,
    val,
    port;
    union REGS
    {
        ir, /* registers send to bios */
        or; /* registers returned from bios */

        ir.h.ah = cmd;
        ir.h.al = val;
        ir.x.dx = port;

        int86(0x14, &ir, &or);

        return or.x.ax;
    }

/* provide a bit of assistance */
help(msg)
char
(
    *msg;

    printf("\n%s\n", msg);

    fputs("install\t\tinstall excom\n", stdout);
    fputs("remove\t\tremove excom\n", stdout);
    fputs("com1\t\t\tsubsequent commands for com1\n", stdout);
    fputs("com2\t\t\tsubsequent commands for com2\n", stdout);
    fputs("dtr\t\t\t\tuse DTR for input flow control\n", stdout);
    fputs("rts\t\t\t\tuse RTS for input flow control\n", stdout);
    fputs("xnxfn\t\t\t\tuse XON/XOFF (^S ^Q) input flow control\n", stdout);
    fputs("nocts\t\t\t\t\trequire CTS to transmit\n", stdout);
    fputs("nodsr\t\t\t\t\trequire DSR to transmit\n", stdout);
    fputs("19200\t\t\t\t\tset baud rate\n", stdout);
    fputs("38400\t\t\t\t\tset baud rate\n", stdout);

    exit(1);
)

```

End Listings

## ARE YOU STUCK?

Stuck with PL/I applications running only on mainframes or minis?

Language Processors, Inc., an innovator in compiler technology, has your solution: a PL/I compiler that rescues your applications from aging mainframes and from proprietary architecture minis.

LPI-PL/I allows you to retain your present PL/I investment by letting you port the applications to just about any UNIX or XENIX-based computer.

LPI-PL/I is a true compiler that produces fast and compact machine code. You can select from several levels of optimization (local, global or machine dependent) to produce efficient applications. And our LPI-DEBUG lets you test and debug programs in PL/I, not machine language.

LPI-PL/I is just one of an entire family of compilers. The LPI family allows you to develop every subprogram of an application in the LPI language best suited for the job, and then execute cross-language calls between the subprograms. You choose the language—we will supply the compiler—with the support that made us stand above the crowd.

Copyright 1987 by Language Processors, Inc. LPI and LPI Logo are trademarks of Language Processors, Inc. The companies mentioned above own numerous registered trademarks.

**LPI: PL/I, COBOL, RPG II, BASIC, FORTRAN, PASCAL, C, DEBUG**

And next time you buy a computer, you won't be stuck—LPI languages will be on it.

LPI-PL/I is a full implementation of ANSI PL/I X3.74-1981 General Purpose Subset. It is compatible with DG and Prime implementations of PL/I and has extensions for compatibility with mainframe and DEC implementations. Some of these extensions are: SELECT, BYTE, LEAVE, %REPLACE.

Available on MC680X0-based computers, such as Altos, Apollo, Arete, AT&T, CT, NCR, Sun and Unisys.

NEW: on AT&T 3B2, 3B5 and 3B15 Series computers.

NEW: on Intel 80386-based computers.

Call the LPI Retail Group now to order, or send the coupon. Prices start at \$995 for 80386 machines.

I need more information on LPI-PL/I for the following computers:

I need more information on your other language(s):

Name Title (or attach your business card)

Address Phone



Language Processors, Inc.  
400-1 Totten Pond Road  
Waltham, MA 02154  
Telephone: 617-890-1155  
Telefax: 617-890-5633  
Telex: 951-671



## FORTH DEVELOPMENT ENVIRONMENT

Take advantage of Bryte's tools to make your job easier:

- Bryte's development environment uses BRYTE-FORTH on the actual production hardware during product development. No emulators, no changes, no surprises.
- Optional PC-based cross-development tools use DOS files as microcontroller mass storage. These files can be used to generate compact EPROM images, de-tailed listings, and cross-references.

Why not start developing the Bryte way today?

BRYTE-FORTH 8031 EPROM	100.00
(includes 130 page User's Manual)	
Utility disk(s)	65.00*
Cross-compiler/Cross-assembler	235.00*
8031 unlimited quan. license	1000.00*

\* Includes complete source code

**bryte computers, inc.**

P.O. Box 46  
Augusta, ME 04330-0046

207/547-3218

CIRCLE 387 ON READER SERVICE CARD



to



**the dBx™ translator**

- **dBx** produces quality **C** direct from dBASE II or III programs.
- Move dBASE programs to UNIX or other machines.
- Improve program speed and reliability.
- Support multi-user/network applications.
- With power guidebook of conversion hints.
- Includes full screen handler and uses your current **C** database manager.
- May be used to move existing programs or help dBASE programmers learn **C** easily.
- For MSDOS, PCDOS, UNIX, XENIX, Macintosh, AMIGA. (Uses ANSI.SYS driver on MSDOS, CURSES under UNIX)
- Priced from \$350, also available from distributors.

dBx is a trademark of

**Desktop Ai**

1720 Post Road E., Westport, CT 06880 MCIMAIL • DESKTOPAI  
Phone • 203-255-3400 Telex • 6502972226MCI

CIRCLE 258 ON READER SERVICE CARD

## TURBO PASCAL OVERLAYS

### Listing One (Text begins on page 50.)

Listing 1: MemOvrly.Inc

```

*****
*                               *
*       Turbo Pascal Memory Overlay Routines                       *
*                               *
*       Copyright (C) 1986 by Steve McMahon                       *
*                               *
*       All Rights Reserved.                                         *
*****

```

(\*

Limitations:

These routines have been tested only for Turbo 3.01A (both PC-DOS and generic MS-DOS). They may not work under 3.0 (the celebrated FileSize bug may cause trouble) and will certainly not work under 2.0XX.

Memory overlay files must be < 64k in size!

NORMAL overlays nested inside memory overlays should work, but trying to nest memory overlays inside memory overlays would be disastrous!

OvrPath will not work in conjunction with memory overlays! (Writing a replacement routine would be simple if the code below makes sense to you.)

I/O testing in InitOverlay is just Turbo's Native. Anyone really needing memory overlays will probably wish to install their own I/O error checking.

\*)

CONST

```

RequiredHeap = $1000; (Paragraphs of Heap Required by Program
                       for other purposes than memory overlays.
                       Change this to suit your needs for
                       dynamic storage.)

```

TYPE

```

(Type used in both InitOverlay and DisposeOverlayStorage)
OverlayProcedure = RECORD

```

```

CASE Boolean OF

```

```

  True :

```

```

    ( OldCall : ARRAY[1..3] OF Byte;
      OldOffset : Integer;
      FileName : ARRAY[1..13] OF Char;
    );

```

```

  False :

```

```

    ( NewCallInstruction : ARRAY[1..3] OF Byte;
      NewCallAddress : Integer;
      CurrentOffset : Integer;
      OverlayCodeLoc : ^Byte;
      NewRoutineLoc : Integer;
      OverlaySize : Integer;
    );

```

```

END;

```

PROCEDURE NewOverlayHandler;

BEGIN

  INLINE(

    (When this routine receives control, AX contains the number of bytes in the desired overlay & BX contains the offset (in pages) of the desired overlay within the overlay file (now on the heap).)

    (First, check to see if the desired overlay is already in place by comparing DX with the offset recorded in memory immediately after the call instruction. If they match, no load is necessary)

```

$5E/ (POP SI )
$2E/$3B/$14/ (CMP DX,CS:[SI] )
$74/$1B/ (JZ RUN_OVERLAY)

```

  (Save vital registers)

```

$56/ (PUSH SI )
$1E/ (PUSH DS )

```

  (Load ES:DI with destination address (the point the code will run at). Displace to account for header.)

```

$0E/ (PUSH CS )
$07/ (POP ES )
$8B/$FE/ (MOV DI,SI )
$83/$C7/$0D/ (ADD DI,0DH )

```

  (Fetch heap address of source overlay code from memory position two bytes after first byte after call to this routine. Store it in DS:SI)

```

$46/ (INC SI )
$46/ (INC SI )
$2E/$C5/$34/ (LDS SI,CS:[SI] )

```

  (Multiply overlay page by 100H to get number of bytes code is displaced from start of overlay code area (on heap). Add to source offset in SI.)

```

$8A/$F2/ (MOV DH,DL )
$32/$D2/ (XOR DL,DL )
$03/$F2/ (ADD SI,DX )

```

  (Put number of bytes to move in CX)

```

$8B/$C8/ (MOV CX,AX )

```

  (Copy CX bytes from DS:SI to ES:DI)



```

$FC/          (CLD      MOVSB  )
$F3/$A4/      (REP2     )

{Recover mauled registers}
$1F/          (POP      DS      )
$5E/          (POP      SI      )

{RUN OVERLAY;}
$83/$C6/$0D/  (ADD      SI,0DH  )
$FF/$E6       (JMP      SI      )
};
END;

```

```

PROCEDURE InitOverlay(OverlayCallOffset : Integer);
VAR
  OverlayCallPtr : ^OverlayProcedure;
  TestSize, i    : Integer;
  s              : STRING[13];
  f              : FILE;
BEGIN
  OverlayCallPtr := Ptr(CSeg, OverlayCallOffset);
  WITH OverlayCallPtr^ DO
    BEGIN
      {Obtain overlay file name}
      i := 1;
      s := '';
      WHILE FileName[i] <> #0 DO
        BEGIN
          s := s + FileName[i];
          i := i + 1;
        END;
      {Open overlay file as untyped file}
      Assign(f, s);
      Reset(f);
      {determine file size in $80-byte sectors}
      TestSize := FileSize(f);
      {Check to see if there's enough space on the heap.}
      {If there isn't, leave the overlay on disk}
      IF (MemAvail > (RequiredHeap + TestSize * 8)) AND
        (MaxAvail >= TestSize * 8) THEN {there's enough space}
        BEGIN
          {install overlay}
          OverlaySize := TestSize;
          GetMem(OverlayCodeLoc, OverlaySize * $80);
          BlockRead(f, OverlayCodeLoc^, OverlaySize, i);
          NewCallInstruction[1] := $2E; {CS;}
          NewCallInstruction[2] := $FF;
          NewCallInstruction[3] := $16; {indirect near call}
          NewCallAddress := Ofs(NewRoutineLoc);
          NewRoutineLoc := Ofs(NewOverlayHandler) + 7;
          {extra 7 bytes skips turbo's procedure overhead}
          CurrentOffset := $FFFF; {force load on first call}
        END;
      Close(f);
    END;
  END;
END;

PROCEDURE DisposeOverlayStorage(OverlayCallOffset : Integer);
VAR
  OverlayCallPtr : ^OverlayProcedure;
BEGIN
  OverlayCallPtr := Ptr(CSeg, OverlayCallOffset);
  WITH OverlayCallPtr^ DO
    IF NewCallInstruction[3] = $16 THEN {Overlay is in memory}
      FreeMem(OverlayCodeLoc, OverlaySize * $80);
  END;
END;

```

End Listings

# ATTENTION

## C-PROGRAMMERS

### File System Utility Libraries

All products are written entirely in K&R C. Source code included, No Royalties, Powerful & Portable.

#### BTree Library

**75.00**

- High speed random and sequential access.
- Multiple keys per data file with up to 16 million records per file.
- Duplicate keys, variable length data records.

#### ISAM Driver

**40.00**

- Greatly speeds application development.
- Combines ease of use of database manager with flexibility of programming language.
- Supports multi key files and dynamic index definition.
- Very easy to use.

#### Make

**59.00**

- Patterned after the UNIX utility.
- Works for programs written in every language.
- Full macros, File name expansion and built in rules.

Full Documentation and Example Programs Included.

ALL THREE PRODUCTS FOR — **149.00**

For more information call or write:

**softfocus**

Credit cards accepted.

1343 Stanbury Drive  
Oakville, Ontario, Canada  
L6L 2J5  
(416) 825-0903

Dealer inquiries invited.

CIRCLE 259 ON READER SERVICE CARD

### A PROGRAMMER'S TOOL BOX

## SCREEN MASTER®

ONLY  
**\$99<sup>95</sup>**

A DP MANAGER'S  
BEST FRIEND

**PURE GENIUS IS NOT ENOUGH**  
(YOU STILL NEED THE RIGHT TOOLS)

- DESIGN MENUS
- CREATE PROTOTYPES
- CREATE TUTORIALS
- CAPTURE SCREENS
- CREATE DEMOS
- RUN TIME MODULE

ENHANCE HIGH LEVEL LANGUAGES WITH FULL  
ACCESS TO ALL CAPABILITIES IN YOUR OWN CODE

"source code was reduced by one third..."

"15 minutes to design a sophisticated screen..."

Scott McCaffrey, Musco of PA

"In a word, fantastic..."

"...I just returned my copy of Dan Bricklin's

Demo Program." Thomas Emr, Dir. of Marketing - ADP Inc.

**GENESIS**  
**DATA SYSTEMS**

5403 Jonestown Rd., Harrisburg, PA 17112  
(717) 652-1200

CIRCLE 373 ON READER SERVICE CARD



# MetaWINDOW

## Power Graphics for your PC!

### PC TECH JOURNAL

#### "Product of the Month"

"... a technological tour de  
force for fast PC araphics."

#### NO ROYALTIES!

MetaWINDOW is the advanced high  
performance graphics toolkit which  
breaks the barrier between low-level  
graphic primitive libraries and  
pre-packaged window managers.

### PC MAGAZINE

"... the only way I know to do  
advanced graphics from Turbo  
Pascal."  
"... testing has found it fast and  
reliable."

#### Jeff Duntemann

"MetaWINDOW is written in  
hand optimized assembly code  
and is very fast. It is virtually  
bug free, and the manufacturer  
has been very responsive to  
questions. If you intend to do  
any graphics work at all, you  
must have this product."

### PC TECH JOURNAL

"... solves many problems that  
bedevil graphics sotware develop-  
ers: non-standard hardware, high  
library cost, poor documentation,  
license fees, and confining  
license agreements. For proprie-  
tary graphics, this product may  
be the only choice."

"... out-Borlands Borland's own  
Turbo Graphics Toolbox."

"... one high-powered piece of  
artillery."

MetaWINDOW includes over 200+  
graphic drawing and windowing functions  
and comes complete with langauge  
bindings for 15 popular C, Pascal and  
Fortran compilers, plus dynamic runtime  
support for over 40 graphics adaptors  
and input devices.

#### MetaWINDOW

Advanced Graphics Windowing Toolkit  
4 disks, 3 260 page manuals - \$185

#### TurboWINDOW

All the features of MetaWINDOW  
just for Turbo Pascal - \$79.50

## METAGRAPHICS

SOFTWARE CORPORATION

269 Mount Herman Road  
Scotts Valley, CA 95066  
408/438-1550

CIRCLE 392 ON READER SERVICE CARD

# UNIX BBS

## Listing One (Text begins on page 54.)

```
list

1  TPATH=/u/bbs/rbin
   # check the command argument - $1
2  if [ "$1" = MS ]
3  then
   # argument is for MS-DOS files
4      echo 'who am i | cut -f1 -d" " 'date | cut -c1-16 ' "MS.list" >>/u/bbs/log.file
5      echo
6      more $TPATH/MS.list
7  elif [ "$1" = Mac ]
8  then
   # argument is for Macintosh files
9      echo 'who am i | cut -f1 -d" " 'date | cut -c1-16 ' "Mac.list" >>/u/bbs/log.file
10     echo
11     more $TPATH/Mac.list
12  elif [ "$1" = Unix ]
13  then
   # argument is for Unix files
14     echo 'who am i | cut -f1 -d" " 'date | cut -c1-16 ' "Unix.list" >>/u/bbs/log.
                                     file
15     echo
16     more $TPATH/Unix.list
17  else
   # no argument was entered
18     echo
19     echo "List which directory? (MS, Mac, or Unix)"
20  fi
```

End Listing One

## Listing Two

```
dwld

   # identify which file directory (contained in the first argument - $1)
1  if [ "$1" = MS ]
2  then
   # an MS-DOS file
3      filedir=/u/bbs/MS-files
4  elif [ "$1" = Mac ]
5  then
   # a Macintosh file
6      filedir=/u/bbs/Mac-files
7  elif [ "$1" = Unix ]
8  then
   # a Unix file
9      filedir=/u/bbs/Unix-files
10 else
   # no valid file directory was entered
11     echo "Follow the dwld command with a file directory - MS, Mac, or Unix"
12  fi

   # verify that a file name has been entered (contained in the second argument - $2)
13 if [ -n "$2" ]
14 then
   # verify that the file name exists within the selected directory
15     if [ -f "$filedir/$2" ]
16     then
17         echo 'who am i | cut -f1 -d" " 'date | cut -c1-16 ' "dwld" $1 $2
                                     >>/u/bbs/log.file

   # select a file transfer protocol
18     echo "Transfer potocol (X = Xmodem; A = ASCII): \c"
19     read method
20     if [ "$method" = X -o "$method" = x ]
21     then
   # send a binary file using the XModem protocol
22         xmodem -sb $filedir/$2
23     else
   # send an ASCII file
24         echo "Prepare to transfer file. Press return to start."
25         read dummy
26         cat $filedir/$2
27         sleep 5
28         echo "Download complete. Press return to continue."
```



```

29      read dummy
30      fi
31      else

# a valid file name wasn't entered

32      echo "Sorry, that file name does not exist. Be sure to type the"
33      echo "file name exactly as it appears in the directory listing."
34      echo "Upper and lower case letters are different."
35      fi
36      else

# no file name was entered

37      echo "Please enter a file name after the directory on the command line."
38      echo "Use the list command to see available file names."
39      fi

```

**End Listing Two**

### Listing Three

```

upld

# verify that a file name has been entered (in argument 1 - $1)
1  if [ -n "$1" ]
2  then

# verify that a file with the same name isn't already stored in the Uploads directory
3      if [ -f /u/bbs/Uploads/$1 ]
4      then

# the file name already exists
5      echo "Please give your file another name. $1 already exists."
6      else

# collect the file transfer protocol
7      echo 'who am i | cut -f1 -d" " ' `date | cut -c1-16` "upld" $1 >>/u/bbs/log.
                                                file
8      echo "Transfer protocol (X = Xmodem; A = ASCII): \c"
9      read method
10     if [ "$method" = X -o "$method" = x ]
11     then

# receive a binary file using the XModem protocol
12         xmodem -rb /u/bbs/Uploads/$1
13     else

# receive an ASCII file
14         echo
15         echo "Begin sending ASCII file."
16         echo "Type a CNTRL-d when finished transmitting."
17         echo
18         cat > /u/bbs/Uploads/temp
19         cp /u/bbs/Uploads/temp /u/bbs/Uploads/$1
20     fi

# collect the file description
21     echo "Please enter a one line description of your file."
22     read desc
23     echo $1 $desc >>/u/bbs/Uploads/Doc.file
24     fi
25     else
# no file name was entered

26     echo "Please enter the name of the file you wish to upload on"
27     echo "the command line."
28     fi

```

**End Listing Three**

### Listing Four

```

readmsg

1  FIRST=/u/bbs/.first
2  LAST=/u/bbs/.last
3  MSG=/u/bbs/msg
4  echo 'who am i | cut -f1 -d" " ' `date | cut -c1-16` "readmsg" >>/u/bbs/log.file
5  echo

# display the number of the first and last messages available
6  echo "Messages available:"
7  cat $FIRST
8  echo "through"
9  cat $LAST
10 echo

# accept the number of the first message to be displayed
11 echo "Message number to read ('q' to quit): \c"
12 read message

# loop until the user wants to quit
13 while [ "$message" != q ]
14 do
15     echo

# verify that the message number entered actually exists

```

*(continued on next page)*

## Fortran Support for IBM PC/XT/AT & Compatibles

### Versions Available For:

Microsoft, Supersoft, RyanMcFarland,  
IBM Professional, Lahey, & IBM  
Fortran.

### Forlib-Plus \$69.95

Supports graphics, interrupt driven communication, program chaining, and file handling/ disk support. A Fortran coded subroutine is included which will plot data on the screen either in linear/linear, log/linear, linear/log, or log/log on the appropriate grid.

### Strings & Things \$69.95

Supports string manipulations, command line usage, DOS call capabilities, SHELL generation and data transmission, BATCH file control, music generation, PEEKS and POKES, PORT access, and general register manipulations.

### For-Winds \$89.95

Gives the Fortran programmer the capability of generating up to 255 windows on the screen. Each window can be individually scrolled, moved, sized, generated, and removed. Both color and monochrome type displays are supported. Full source code is supplied for customization.

### ACS Time Series \$495.00

This is a COMPLETE time series analysis package which contains VERY HIGH SPEED FFTs, Filter generations, convolutions, transfer function calculations, auto and cross spectra calculations, Cepstrum, curve fitting algorithms, coherence calculations, and many other associated routines. The price includes FULL source code.

### Fortran Scientific Subroutine Package \$295.00

There are approximately 100 Fortran subroutines included which fall under the following 12 categories:

1) Matrix storage and Operations 2) Correlation and Regression, 3) Design Analysis (ANOVA), 4) Discriminant Analysis, 5) Factor Analysis, 6) Eigen Analysis, 7) Time Series, 8) Nonparametric Statistics, 9) Distribution Functions, 10) Linear Analysis, 11) Polynomial Solutions, 12) Data Screening. Full source code is included.



**ALPHA COMPUTER SERVICE**  
5300 ORANGE AVENUE SUITE 108  
CYPRESS, CALIFORNIA 90630  
(714) 828-0286

California Residents  
Include 6% Sales Tax      There are NO license fees

**CIRCLE 321 ON READER SERVICE CARD**



# MULTITASKING

## Introducing MultiDos Plus

The new multitasking software  
for the IBM-PC.

*Ideal* for developing applications in process control, data acquisition, communications, and other areas. Check these features which make **MultiDos Plus** an unbeatable value.

- Run up to 32 programs concurrently.
- Your software continues to run under DOS. No need to learn a new operating system.
- Use the compilers you already have. Supports software written in any language.
- Operator commands to load/run programs, change priority, check program status, abort/suspend/resume programs.
- Programmatic interface via INT 15H for the following.
  - \* Intertask message communication. Send/receive/check message present on 64 message queues.
  - \* Task control by means of semaphores. Get/release/check semaphores.
  - \* Change priority-128 priority levels.
  - \* Suspend task for specified interval.
  - \* Spawn and terminate external and internal tasks.
  - \* Disable/enable multitasking.
  - \* and more!
- Independent foreground/background displays.
- Access to DOS while applications are running.

### Hardware/Software Requirements

IBM PC/XT/AT or true clone. Monochrome/CGA display adaptors or equivalent cards only. Enough memory to hold **MultiDos Plus** (48 KB) and all your application programs. Also may need 4 or 16 KB memory for "hidden screens" for each active task. MS-DOS (or PC-DOS) 2.0 or later operating system.

ONLY

# \$49.95

Outside USA add \$5.00 shipping and handling.

Visa and Mastercard orders call toll-free: 1-800-367-6707. In Mass call 617-651-0091, or send check or money order to:

## NANOSOFT

13 Westfield Rd, Natick, MA 01760

MA orders add 5% sales tax. Write for source code and quantity price.

CIRCLE 309 ON READER SERVICE CARD

# UNIX BBS

## Listing Four (Listing continued, text begins on page 54.)

```
16 if [ -f $MSG/$message ]
17 then
    # display the message
18     cat $MSG/$message
19 elif [ "$message" != q ]
20 then
    # the entered message number doesn't exist
21     echo "Sorry. That message doesn't exist."
22 fi
23 echo
    # accept the next message number to read
24 echo "Message number to read: ('q' to quit): \c"
25 read message
26 done
27 echo
```

End Listing Four

## Listing Five

```
send

1 echo 'who am i | cut -f1 -d" " 'date | cut -c1-16' "send" >>/u/bbs/log.file
2 SCAN=/u/bbs/msg/.index
3 LAST=/u/bbs/.last
4 MSG=/u/bbs/msg
5 L='cat $LAST'
6 D='date'

    # increment the number of the last message entered
7 L='expr $L + 1'

    # save the new "last message" value
8 echo $L >$LAST
9 echo

    # collect message header information
10 echo "To: \c"
11 read to
12 echo "Subject: \c"
13 read subject
14 echo "From: \c"
15 read who
16 trap 'rm -f $MSG/$L; continue' 2 3
17 ech
18 echo "Start typing your message. You have 20 lines available."
19 echo "Type a period on a new line to end the message."
20 echo

    # store the message header in the message file
21 echo "# $L From: $who $D" >$MSG/$L
22 echo " To: $to" >>$MSG/$L
23 echo " Subject: $subject" >>$MSG/$L
24 echo >>$MSG/$L

    # collect the body of the message
25 NL=0
26 read newline
27 while [ "$newline" != "." -a "$NL" -lt 20 ]
28 do
29     echo $newline >>$MSG/$L
30     NL='expr $NL + 1'
31     read newline
32 done
33 echo

    # update the message index for use by the scan command
34 echo "Message being posted, please wait... \c"
35 echo "# $L From: $who $D" >>$MSG/temp
36 echo " To: $to" >>$MSG/temp
37 echo " Subject: $subject" >>$MSG/temp
38 echo " " >>$MSG/temp
39 cat $SCAN >>$MSG/temp
40 mv $MSG/temp $SCAN
41 echo
```

End Listing Five

## Listing Six

```
pmail

1 USERS=/u/bbs/rbin/user.file
2 echo 'who am i | cut -f1 -d" " 'date | cut -c1-16' "pmail" >>/u/bbs/log.file

    # display pmail instructions
3 echo
4 echo" ***** Private Mail *****"
5 echo
6 echo      Commands:
7 echo      s - send mail to another user
```

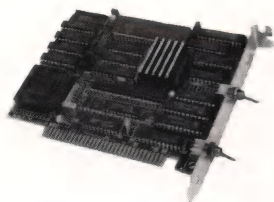
(continued on page 84)



# MICROWAY ACCELERATES YOUR PC!

## FastCACHE-286™

**Runs your PC Faster than an AT!**  
Runs the 80286 at 9 or 12 MHz and the 80287 at 8, 9 or 12 MHz. Includes 8 kbytes of 55ns CACHE



Compatible with Leading Edge Model D, Compaq, and Turbo motherboards. Includes 8088 Reboot Switch, DCache, Print Spooler and Diagnostics... **From \$399**

## LOTUS/INTEL EMS SPECIFICATION BOARDS

**MegaPage™** The only EMS board which comes populated with two megabytes of cool-running, low power drain CMOS RAM installed. Includes RAM disk, print spooler, disk cache and EMS drivers. For the IBM PC, XT and compatibles... **\$549**

**MegaPage with 0K..... \$149**

**MegaPage with 2 megabytes of HMOS RAM..... \$419**

**MegaPage AT/ECC™** EMS card for the PC AT and compatibles includes Error Correction Circuitry. With ECC, 11 RAM chips cover 256K so the user never encounters RAM errors. With 1 megabyte CMOS RAM..... **\$699**

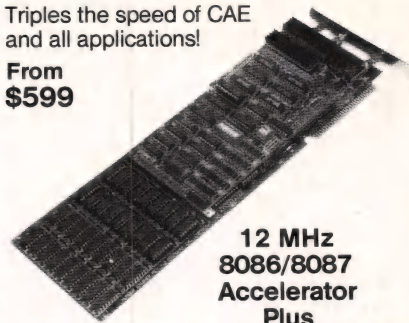
**INTEL, JRAM, or Maynard .... CALL**

**INTEL INBOARD 386 0K..... \$1325**

## NUMBER SMASHER/ECM™

Triplies the speed of CAE and all applications!

**From \$599**



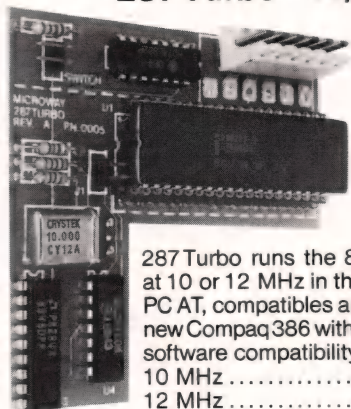
**12 MHz  
8086/8087  
Accelerator  
Plus  
A Megabyte for DOS!**

For the IBM PC, XT and compatibles  
**PC Magazine "Editor's Choice"**

## 8087 SOFTWARE

IBM BASIC COMPILER.....	<b>\$465</b>
MICROSOFT QUICK BASIC.....	<b>\$79</b>
87BASIC COMPILER PATCH.....	<b>\$150</b>
87BASIC/INLINE.....	<b>\$200</b>
IBM MACRO ASSEMBLER.....	<b>\$155</b>
MS MACRO ASSEMBLER.....	<b>\$99</b>
87MACRO/DEBUG.....	<b>\$199</b>
MICROSOFT FORTRAN V4.....	<b>\$299</b>
RM FORTRAN.....	<b>\$399</b>
LAHEY FORTRAN F77L.....	<b>\$477</b>
MS or LATTICE C.....	<b>CALL</b>
STSC APL★PLUS/PC.....	<b>\$450</b>
STSC STATGRAPHICS.....	<b>\$675</b>
SPSS/PC+.....	<b>\$695</b>
87SFL Scientific Functions.....	<b>\$250</b>
87FFT.....	<b>\$200</b>
OBJ - ASM.....	<b>\$200</b>
PHOENIX PRODUCTS.....	<b>CALL</b>

## 287 Turbo™ -10/12



287 Turbo runs the 80287 at 10 or 12 MHz in the IBM PC AT, compatibles and the new Compaq 386 with 100% software compatibility.

10 MHz..... **\$450**  
12 MHz..... **\$550**

**PC Magazine "Editor's Choice"**

## 8087 UPGRADES

All MicroWay 8087s include a one year warranty, complete MicroWay Test Program and installation instructions.

**8087 5 MHz..... \$105**

For the IBM PC, XT and compatibles

**8087-2 8 MHz..... \$154**

For Wang, AT&T, DeskPro, NEC, Leading Edge

**80287-3 5 MHz..... \$179**

For the IBM PC AT and 286 compatibles

**80287-6 6 MHz..... \$229**

For 8 MHz AT and compatibles

**80287-8 8 MHz..... \$259**

For the 8 MHz 80286 accelerator cards and Compaq 386

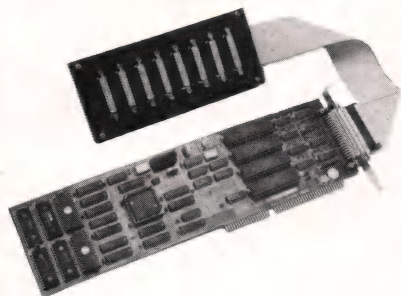
**80287-10 10 MHz..... \$395**

**PC-PAL™ Programmer..... \$395**

Call for great prices on V20, V30, 64K, 128K and 256K RAM

## AT8™

Turns your AT into a high speed, multi-user Xenix business system!



8 port, intelligent serial controller with 3% response degradation. Includes 8 MHz 80186 with built in DMA..... **\$1299**

## MICROWAY SOFTWARE FOR LOTUS 1-2-3™

**FASTBREAK™** employs the 8087 to increase the speed of Lotus 1-2-3™ Version 1A or 1A\*. Users are reporting speed ups of between 3 and 36 to 1. When run with our NUMBER SMASHER accelerator card, recalculation speed ups of 10 to 30 are being reported..... **\$79**

**PowerDialer®** Add-In for Lotus 1-2-3 Release 2. Automated telephone dialing from within 1-2-3. Adds least cost routing, automatic carrier selection and automated phone book worksheet. Builds customized dialing applications. Can be used with DesqView..... **\$79**

**HOTLINK™** adds easy linking of spreadsheets to Lotus 1-2-3 Version 1A... **\$99**

## 287 TURBO-PLUS™

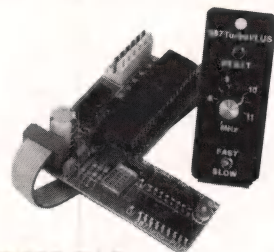
**Speeds up your AT**

Adjustable 80286 Clock 6-12 MHz

10 MHz 80287 Clock

Plus Full Hardware Reset..... **\$149**

Optional 80286-10..... **\$175**



**287TURBO-PLUS**

With 80287 10 MHz..... **\$549**

With 80287 12 MHz..... **\$629**

**CALL (617) 746-7341 FOR OUR COMPLETE CATALOG**

**MicroWay** P.O. Box 79  
Kingston, Mass.  
02364 USA  
(617) 746-7341

**You Can  
Talk To Us!**

**MicroWay Europe**  
32 High Street  
Kingston-Upon-Thames  
Surrey England KT1 1HL  
Telephone: 01-541-5466



## Quelo<sup>®</sup> 68000 Software Development Tools

First release 1983 - MOTOROLA compatible - produces ROMable code, S-records, extended TEK hex, UNIX COFF. Portable SOURCE CODE. Native and cross versions on: ATARI ST, AMIGA, Masscomp, Sun, Apollo, Charles River, VAX VMS and UNIX.

### 68020 Cross Assembler Package

Supports 68000, 68010, 68020, 68881 and 68851  
For CP/M-68K and MS/PC-DOS - \$750

### 68000/68010 Cross Assembler Package

For CP/M-80, -86, -68K and MS/PC-DOS - \$595

### 68020 Disassembler

Supports 68000, 68010, 68020, 68881, 68851  
For CP/M-68K and MS/PC-DOS - \$495/295, Amiga and Atari ST - \$119/79,  
CRDS UNOS - \$995/595

### 68000/68010 Software Simulator

For MS/PC-DOS by Big Bang Software, Inc. - \$285, VAX - \$2000

### 68000 "C" Cross Compiler

For MS/PC-DOS by Lattice, Inc. - \$500

Call Patrick Adams today:

Quelo, Inc.  
2464 33rd. West, Suite #173  
Seattle, WA USA 98199  
Phone 206/285-2528  
Telex 910-333-6171

TM Quelo, Quelo, Inc. MS, Microsoft Corporation, CPM, Digital Research

CIRCLE 377 ON READER SERVICE CARD

**DISK FORMAT  
CONVERSION**

PC-DOS program  
lets your PC  
Read/Write/Format  
over 300 formats

**XENOCOPY-PC™**

by Fred Cisin

**\$79.95** + \$5.00 S/H Sales Tax if CA.

Upgrades available from previous versions

Ask about FREE CP/M emulator! ➡

To Order Contact:

**XENOSOFT™**

1454 Sixth Street, Berkeley, CA 94710



(415) 525-3113



CIRCLE 225 ON READER SERVICE CARD

## Dr. Dobb's Journal

Subscription  
Problems?  
No Problem!



Give us a call and we'll  
straighten it out. Today.

Outside California  
CALL TOLL FREE: 800-321-3333

Inside California  
CALL: 619-485-6535 or 6536

## UNIX BBS

### Listing Six (Listing continued, text begins on page 54.)

```

8  echo    r - read mail sent to you
9  echo    l - list the users by user id and name
10 echo    x - exit the private mail system
11 echo

# accept the first pmail command

12 echo "Command: \c"
13 read choice

# loop until the user wants to quit

14 while [ "$choice" != x -o "$choice" != X ]
15 do
16     echo
17     case $choice in
18         [Ss])

# send private mail
# print instructions

19         echo "WARNING: To successfully send mail observe these rules:"
20         echo
21         echo "  1. If you wish to erase a character, use the backspace key."
22         echo "DO NOT, DO NOT use the delete key."
23         echo "  2. Type a carriage return at the end of each line on the"
24         echo "screen. Lines should be no more than 80 characters."
25         echo "  3. Do not use the following characters in your message (they"
26         echo "have special meaning to the system): @ and #"
27         echo
28         echo "Press RETURN to begin sending your message: \c"
29         read dummy
30         echo

# collect user id of recipient

31         echo "To whom (user id): \c"
32         read to

# verify that recipient entered is a valid user id

33         TPERS='grep $to $USERS | wc -c'
34         if [ "$TPERS" -eq 0 ]
35         then

# user id is invalid

36         echo
37         echo "Sorry, that user id doesn't exist."
38         echo "List them with the l command."
39         else

# send mail

40         echo
41         echo "Start typing your message. Type a period on a new"
42         echo "line to send the message."
43         echo
44         mail $to
45         fi

# collect next option

46         echo
47         echo "Command (x to exit): \c"
48         read choice
49         continue;;
50         [Rr])

# read private mail
# print instructions

51         echo "After each message you will receive a '?' prompt."
52         echo "Type 'd' to delete the message, <cr> to leave it in"
53         echo "your mailbox, or 'q' to stop reading mail."
54         echo

# read the mail

55         mail
56         echo

# collect the next command

57         echo "Command (x to exit): \c"
58         read choice
59         continue;;
60         [Ll])

# display a list of user names and associated user id's

61         echo
62         more $USERS
63         echo

# collect the next command

64         echo "Command (x to exit): \c"
65         read choice
66         continue;;
67         *)

# if an unrecognized command was entered, prompt for another one

68         echo "Command (x to exit): \c"
69         read choice
70         continue;;
71     esac
72 done
73 echo

```

End Listing Six

(Listing Seven begins on page 86.)



# EVEN MORE POWER AND FLEXIBILITY BRIEF 2.0

Users and industry press alike have unanimously proclaimed BRIEF as the best program editor available today. Now, the best gets better, with the release of BRIEF 2.0.

Straight from the box, BRIEF offers an exceptional range of features. Many users find that BRIEF is the only editor they'll ever need, with features like real, multi-level Undo, flexible windowing and unlimited file size. But BRIEF has tremendous hidden power in its exclusive macro language. With it, you can turn BRIEF

into your own custom editor containing the commands and features you desire. It's fast and easy.

Jerry Pournelle, columnist for BYTE magazine summed it all up by saying BRIEF is, "Recommended. If you need a general purpose PC programming editor, look no further." His point of view has been affirmed by rave reviews in C JOURNAL, COMPUTER LANGUAGE, DR. DOBB'S JOURNAL, DATA BASED ADVISOR, INFOWORLD AND PC MAGAZINE.

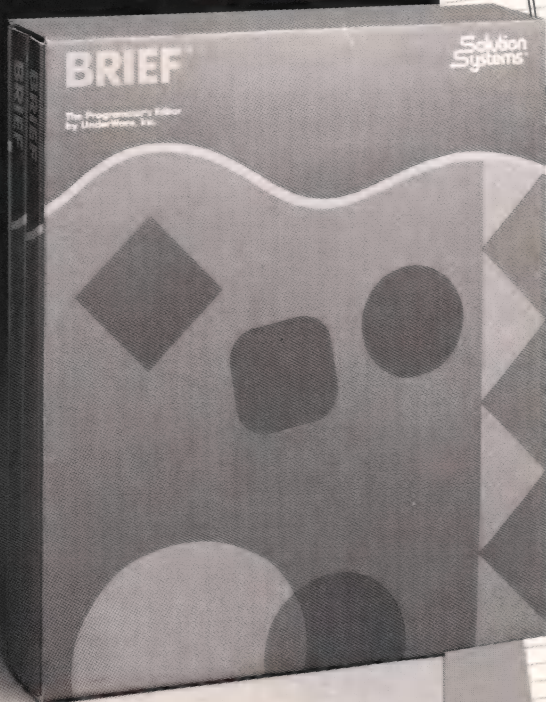
One user stated "BRIEF is one of the few pieces of software that I would dare call a masterpiece." Order BRIEF now and find out why. BRIEF 2.0 is just \$195. If you already own BRIEF, call for upgrade information.

**TO ORDER CALL: 1-800-821-2492  
(in MA call 617-659-1571)**

As always, BRIEF comes with a 30 day money-back satisfaction guarantee.

**Solution  
Systems™**

335 Washington St.  
Norwell, MA 02061  
(617) 659-1571



## Look at these BRIEF 2.0 enhancements!

### Main Features:

- All new documentation with tutorials on basic editing, regular expressions and the BRIEF Macro Language.
- Setup program for easy installation and configuration. (Requires no knowledge of the macro language)
- Increased speed for sophisticated operations like Undo and Regular Expression Search.
- Expanded regular expressions, with matching over line boundaries.
- More block types, with marking by character, line or column.
- Command line editing (move cursor, add and delete characters, specify command parameters).
- Support for more programming languages.
- Optional borderless windows.
- Enhanced large display support, including wider displays.
- Reconfigurable indenting for C files (supports most indenting styles).

### Basic Features:

- Full multi-level Undo
- Windows
- Edit many files at once
- File size limited only by disk space
- Automatic language sensitive indentation

Plus the basic  
features that made  
BRIEF SO popular!

Requires an IBM PC or compatible with  
at least 192K RAM.  
BRIEF is a trademark of UnderWare, Inc.  
Solution Systems is a trademark of Solution Systems.

CIRCLE 142 ON READER SERVICE CARD



Screen  
Designer

Data&Windows, the window-oriented data-entry system from Magus, Inc., now supports more of your favorite compilers! If you use Microsoft C, Pascal, or FORTRAN, Manx Aztec C86, Mark Williams C Programming System, Lattice C, IBM C, MetaWare High C, MetaWare Professional Pascal, or Datalight Optimum C, you can use Data&Windows to quickly and easily create almost any user interface.

With Data&Windows, you draw your text, fields, and colors on the video display. You see exactly what you will get while you create it. The screen designer gives you maximum functionality in an editor-like environment. Fields have options for specifying character and data type validations, user-definable validations, protected (display only) fields, auto-erasing fields, retain data fields, and more. You can check the look and operation of the screen with test mode.

Screens are saved in Microsoft object file format. You simply link the screen files with your application, so screen and field information is stored in your program (.EXE) file rather than in separate data files that must be present at runtime.

You get library routine support for fully overlapping and scrollable windows which use an approach called **static windowing**. This technique eliminates the need for replication of static data in dynamic memory. More than seventy library routines are available to manipulate your windows, control data entry and storage, create window-oriented menus, and more!

**Get Started Quick!** A thorough on-disk tutorial is provided so that you can begin creating useful screens the day you receive the product. Utility programs that document each screen and allow you to prototype (or simulate) your application complete the package.

**Try It Out!** Order your demo disk today. For \$5.00 (refundable when you buy the product), you will receive a copy of the screen generator, the tutorial, and on-disk documentation on the utility programs and library routines.

Data&Windows is \$345, or \$695 with library source code, and certain discounts are available. Be sure to specify which compiler(s) you use when ordering.

Call (713) 665-4109 for more information. Major credit cards accepted.

## REQUIREMENTS:

IBM PC/XT/AT/JR or true compatible, DOS 2.0 or later, at least 128K free RAM, and one of the compilers listed above. Xenix support will soon become available.

IBM, IBM PC, IBM XT, and IBM AT are trademarks of International Business Machines. Microsoft and XENIX are trademarks of Microsoft Corporation. Aztec is a trademark of Manx Software Systems. High C is a trademark of MetaWare.

MAGUS, INC.  
4545 Bissonet Suite #114  
Bellaire, TX 77401

## UNIX BBS

## Listing Seven (Listing continued, text begins on page 54.)

```
.profile from the Account new
```

```
# set default path to an empty rbin directory
1 PATH=/u/bbs/new/rbin

# set level one prompt to remind users how to log off
2 PS1='Type CNTRL-D now... '

# display the welcome message
3 cat msg1
4 echo

# make sure the user really wants an account
5 echo "Do you wish to request a login? (Y/N) \c"
6 read choice

# continue only if the user answers yes
7 if [ "$choice" = y -o "$choice" = Y ]
8 then
9     echo
10    OK="N"

# loop until user is satisfied with data
11 while [ "$OK" != Y -a "$OK" != y ]
12 do

# collect user name, address, etc.
13     echo "Enter your real name: \c"
14     read realname
15     echo
16     echo "Enter the first line of your mailing address: (4 lines are available)"
17     read address1
18     echo
19     echo "Enter the second line of your mailing address: <cr> if none"
20     read address2
21     echo
22     echo "Enter the third line of your mailing address: <cr> if none"
23     read address3
24     echo
25     echo "Enter the fourth line of your mailing address: <cr> if none"
26     read address4
27     echo
28     echo "Enter your voice telephone number: \c"
29     read phone
30     echo

# display the data entered for the user
31     echo "This is what you have entered:"
32     echo
33     echo $realname
34     echo
35     echo $address1
36     echo $address2
37     echo $address3
38     echo $address4
39     echo
40     echo $phone
41     echo

# ask the user to verify the data
42     echo "Is this correct? (Y/N) \c"
43     read OK
44     done
45     OK="N"

# loop until user is happy
46 while [ "$OK" != Y -a "$OK" != y ]
47 do

# collect the account data
48     echo
49     echo "Enter a one word login name: \c"
50     read logname
51     echo
52     echo "Enter an initial password: \c"
53     read initpasswd
54     echo

# display the entered data for the user
55     echo "You have entered:"
56     echo
57     echo $logname
58     echo $initpasswd
59     echo

# ask the user to verify the data
60     echo "Is this correct? (Y/N) \c"
61     read OK
62     done
63     echo

# write the new account data to a holding file
```

(continued on page 88)



# Professional Programming Products

by BCSofT



## FREE

PC-WRITE™ text editor  
with every purchase.

**PROGRAMMERS AND SOFTWARE DEVELOPERS - LOOK AT THESE PRODUCTS!**

**NEW**

### Quick-Tools™ for QuickBASIC™ users!

- A comprehensive library of over 90 subroutines and functions directly CALLable from your QuickBASIC programs.
- SORT routines allow sorting of single and multidimensional arrays.
- Binary search functions.
- Unique screen handling functions allow splitting screens, fast scrolling, special string printing, character attribute control, phantom cursors, etc.
- Keyboard scanning and status calls, with field inputting and decoding. On screen editing of input fields.
- Unique file handling routines.
- FREE Updates and phone support!
- Plus much, much more!

Only \$129.95 Complete

**NEW**

### NET-TOOLS™ NETBIOS Programming Tools

- NET-TOOLS allows you to write programs for ANY NETBIOS compatible local area network - fast and easily. CALLable from Microsoft Assembler, C, PASCAL, or FORTRAN.
- Add and Delete local names.
- Initiate and Cancel sessions. Just give NET-TOOLS the name of the computer you would like to call, and it will automatically locate the user and make the connection for you.
- Transfer Messages with automatic retries and error detection. Both the datagram and session protocols.
- Redirect Local Devices simply and easily with a single function call.
- Plus much MORE !
- Complete SOURCE CODE is provided, FREE !
- FREE Updates and phone support!

ONLY \$149.00 Complete

### TURBO.ASM™ For Turbo PASCAL users !

- A unique programming tool which is a must for every Turbo Pascal user. The only package designed for interfacing assembly language with Turbo Pascal.
- Outlines several different ways to add assembly language routines to Turbo programs, some without effecting your code space.
- Fully explains the internal workings of Turbo Pascal and data passing methods.
- Includes a library of Assembly routines which can be used directly from Turbo Pascal.
- The best way to learn assembly language.
- FREE Updates and phone support!

Only \$99.95 Complete

### ASMLIB™

#### The Programmer's Library

- A set of over 210 subroutines to greatly increase your productivity. Written in assembly language and directly CALLable from Microsoft Assembler, C, PASCAL, and FORTRAN.
- Complete SOURCE CODE provided -FREE!
- Text WINDOWing functions allow up to 64 overlapping windows.
- TERMINATE and STAY RESIDENT programs are written easily. Programs can "POP UP" with a simple keystroke. Use DOS calls from them, also.
- GRAPHICS on the EGA, CGA, and Hercules™ Monochrome.
- Virtual file functions.
- Full FLOATING POINT math and trig with 8087 support.
- Int. driven async. support, plus MUCH MORE!

Only \$149.00 Complete.

**NO ROYALTIES REQUIRED**

To ORDER call or write to:

BC Associates  
A division of BCSofT Corporation  
3261 N. Harbor Blvd.  
Suite B  
Fullerton, CA. 92635



VISA, M/C, or COD orders are welcome!



**CALL TOLL FREE**

**1-800-262-8010**

**in CA. dial 1-714-526-5151**



New  
Release!

# SEIDL MAKE UTILITY Version 2.0

The BEST just got BETTER!

If you're serious about software development, consider the SEIDL MAKE UTILITY (SMK). SMK is not just another copy of the Unix Make. It was specifically designed to deliver features and performance not found in other makes.

✓ **Structured Language** to describe dependencies in a clear, concise and portable manner.

✓ **Rich Command Set** includes parameterized macros, variables, if-then-else, iteration, wild cards, exception cases, macro libraries, interactive statements, environment access, pattern matching and much more!

✓ **Intelligent Analysis** algorithm handles nested include files, library dependencies, and performs consistency tests to detect errors that other makes would blindly ignore.

✓ **Seidl Version Manager** compatibility lets you expand your system into the most comprehensive revision/version control system available.

"SMK is a very good Make indeed. Its major distinction is a truly simple Dependency Definition Language, easy to learn and easy to use... you'll probably bless SMK."

— Sextant, July '86

"SMK offers many unique features. [The error handling facility] is extremely useful if a large number of files must be recompiled."

— Computer Language, June '86

DOS Version Only **\$99<sup>95</sup>** \$3.50 p&h  
Call for other op systems.

**Call Today**  
1-313-662-8086

Visa/MC/COD Accepted  
Dealer Inquiries Invited

**SEIDL COMPUTER ENGINEERING**

3106 Hilltop Dr., Ann Arbor, MI 48103

CIRCLE 114 ON READER SERVICE CARD

## UNIX BBS

### Listing Seven (Listing continued, text begins on page 54.)

```
64      echo $realname >>signups
65      echo $address1 >>signups
66      echo $address2 >>signups
67      echo $address3 >>signups
68      echo $address4 >>signups
69      echo $phone >>signups
70      echo $logname >>signups
71      echo $initpasswd >>signups
72      echo >>signups

# tell the user that he or she is finished

73      echo "This completes the login request process. You'll receive an account"
74      echo "confirmation through the mail within a week."
75      echo
76 fi

# tell the user to log off

77 echo
78 echo "Enter CTNRL-D to log off the system."
79 echo
```

End Listing Seven

### Listing Eight

```
.profile for the info Account

# set the default path to an empty rbin directory

1 PATH=/u/bbs/info/rbin

# set the level one prompt to the logoff message

2 PS1='Type CNTRL-D now...'

# display the welcome message

3 echo
4 echo "                Welcome to Scholastech Telecommunications"
5 echo "                Scholastech Info"
6 echo
7 cat msg
8 echo
9 echo "Press the carriage return to begin: \c"
10 read dummy

# display the contents of the information file

11 more info.file
12 echo
13 echo "Press the carriage return when done: \c"
14 read dummy
15 echo

# give the user the choice of whether or not to sign up for workshops

16 echo "Do you wish to sign up for a workshop? (Y/N) \c"
17 read dummy

# loop until user doesn't want to register for any more workshops

18 while [ "$dummy" = y -o "$dummy" = Y ]
19 do
20     echo
21     echo "Enter your name: \c"
22     read regname
23     echo
24     echo "For which workshop are you registering? \c"
25     read workshop
26     echo
27     echo
28     echo "Enter your school or company name: \c"
29     read school
30     echo
31     echo "For how many people are you registering? \c"
32     read people
33     echo
34     echo "Enter a voice phone number where you can be reached in case"
35     echo "there are any questions about your registration: \c"
36     read phone
37     echo

# write the registration data to disk

38     echo $regname >>signups
39     echo $workshop >>signups
40     echo $school >>signups
41     echo $people >>signups
42     echo $phone >>signups
43     echo >>signups

# allow the user to register for more than one workshop

44     echo "Do you wish to register for another workshop? (Y/N) \c"
45     read dummy
46     done

# tell the user how to log off

47 echo
48 echo "Type CNTRL-D to log off the system."
49 echo
```

End Listing Eight

(Listing Nine begins on page 90.)



# HOT GRAPHICS PACKAGE FOR C PROGRAMS.\* \$39.95

**E**verything you need to write dramatic graphics effects into your Eco-C88 C programs. Some of the features include:

- Support for EGA, CGA, and Z100
- Over 100 graphics and support functions, many of which are PLOT-10 compatible.
- Many low level support routines reside outside your small model code-data area
- Can write dots thru the BIOS (for compatibility) or to memory (for speed)
- Graphics function help from CED editor available
- World, pixel or turtle color graphics modes
- 47 standard fill patterns, 17 line dashing patterns, Hershey fonts, plus user defineable fill, dash and fonts
- Supports view areas, rotateable fonts, clipping, arbitrary fill areas, extensive error checking, examples, and user's manual.

A must for the graphics enthusiast and a bargain at only

## \$39.95

\*Requires Eco-C88 C Compiler.

## NEW POP-UP WINDOWS FOR YOUR C PROGRAMS.

This windowing library allows you to add pop up windows in your C programs quickly and easily. Use them for help windows, selection menus, error messages, special effects—anywhere you need an attention getter. Just some of the features include:

- CGA, EGA, and monochrome support
- Slow mode option for "flicker" displays
- Control any program that goes through the BIOS

- Use up to 255 windows
- No special window commands; use print f ()
- Resize and move windows
- Custom window titles and borders
- Can be used with ANSI device driver
- Most of window's code-data lies outside small model limits
- Use any of the IBM text or block characters
- User's manual and examples

The Windowing Library requires an IBM PC compatible BIOS and the Eco-C88 C compiler.

## ONLY \$29.95

## HANDY LIBRARIAN MAKES LIFE EASIER.

Now you can combine your modules, functions, and subroutines into your own library for easy link commands. Fully compatible with ANY standard OBJ format files (not just Ecosoft's products).

With the Ecosoft librarian, you can:

- Add, delete, and extract from a library
- Get table of contents or index of a library
- Combine libraries, control library page size, use switches for combinations, process complex library requests, use wildcards, and do library directives from command files.
- Complete with user's manual

A valuable addition for any programmer.

## ONLY \$29.95

Orders only:

1-800-952-0472

Technical Information:

(317) 255-6476

## THE FIRST PROFESSIONAL 'C' COMPILER FOR UNDER \$60.

A C compiler with many ANSI enhancements at an unbelievably low price. The Eco-C88 C compiler has:

- Prototyping (the new type-checking enhancement)
- Enum and void data types
- Structure passing and assignment
- All operators and data types
- A standard library with more than 200 functions (many of which are System V compatible for greater code portability)
- CC and mini-make that all but automates the compile process
- 8087 support (we sense the 8087 at runtime — no dual libraries)
- ASM or OBJ output for use with MSDOS linker
- Tiered error messages — enable-disable lint-like error checking
- Fast compiles and executing code
- Expanded user's manual
- CED full-screen program editor

Everything you need at the unbelievable price of \$59.95.

Eco-C88 C compiler requires an IBM PC, XT, or AT (or compatible) with 256K of memory, 2 disk drives and MSDOS 2.1 or later.

**NOT COPY  
PROTECTED.**

Ecosoft Inc.

6413 N. College Ave.  
Indianapolis, IN 46220

ORDER FORM CLIP & MAIL TO: Ecosoft Inc., 6413 N. College Ave., Indianapolis, IN 46220

ITEM	PRICE	QTY	TOTAL
Flexi-Graph Graphics	\$39.95		
Window Library	\$29.95		
Eco-Lib Librarian	\$29.95		
Eco-C88 C Compiler CED	\$59.95		

SHIPPING

TOTAL (IND. RES. ADD 5% TAX)

PAYMENT:

☐ VISA

☐ MC

☐ AE

☐ CHECK

CARD #

EXPIR DATE

NAME

ADDRESS

CITY

STATE

ZIP

PHONE

CIRCLE 89 ON READER SERVICE CARD

# ECOSOFT





# You could buy from our competitors . . .

CSS PC/VI <sup>™</sup> (vi, ex)	\$149.00
Micro C-Tools <sup>™</sup> (cal, dc, detab, entab, fgrep, show, sort, uniq, find, od, file, touch, split)	24.95
Dr. Dobb's <sup>™</sup> Shell (sh)	29.95
Lattice <sup>™</sup> Text Management Utilities (grep, diff, ed, wc)	120.00
PC Profiler (prof)	125.00
	<hr/> \$448.90

**OR**

you could order the  
**MKS Toolkit** and get all  
these for *only* \$139.

**BUT WAIT!  
THERE'S MORE!**

You also get *awk*, a 4gl  
report language.

**BUT THAT'S NOT  
ALL!**

. . . and database  
utilities such as *join*,  
*cut*, *paste*

and there is still  
more: over 100  
utilities in all!

such as:

cmp, cpio, ctags, dd, df, ls,  
nm, pg, sed, size, strings,  
time, tr . . .

**MKS Toolkit  
beats the  
competition  
COLD!**

Call us today:  
(519) 884-2251

Not available on 8-track or cassettes.

CIRCLE 249 ON READER SERVICE CARD

## UNIX BBS

### Listing Nine (Listing continued, text begins on page 54.)

```
logs

# create a temporary file with a log on record for each remote user
1  who /etc/wtmp | grep ph >/u/user.log
# if an "a" was entered as an argument to the command, display the entire log files
2  if [ "$1" = a ]
3  then
4      echo
5      echo
6      echo "System Logins"
7      echo

# display all system use since /etc/wtmp was last purged
8      more /u/user.log
9      echo
10     echo "Next?"
11     read dummy
12     echo

# display all BBS use since /u/bbs/log.file was last purged
13     echo "BBS User Log"
14     echo
15     more /u/bbs/log.file
16     echo

# otherwise, show only the last ten entries in each log file
17 else
18     echo
19     echo
20     echo "Last System Logins"
21     echo
22     tail /u/user.log
23     echo
24     echo
25     echo "Last BBS Activity"
26     echo
27     tail /u/bbs/log.file
28     echo
29 fi

# remove the temporary file
30 rm /u/user.log
```

**End Listing Nine**

### Listing Ten

```
usage

# print headings to report file
1  echo "User      Date      Login      Logout" >usage.temp
2  echo "-----" >usage.temp
3  echo " " >>usage.temp

# get a log in record from /etc/wtmp for remote users on device ph1
4  who /etc/wtmp | grep ph1 >temp1

# get the user name
5  cut -f1 -d" " temp1 >temp2

# get the log in date and time
6  cut -c25-36 temp1 >temp3

# get log out record from /etc/wtmp and extract the log out time
7  who -d /etc/wtmp | grep ph1 | grep -v LOGIN | grep -v rc | cut -c32-36 >temp4

# paste together for the report
8  paste temp2 temp3 temp4 >>usage.temp

# print the report
9  pr usage.temp >usage.summary

# remove temporary files
10 rm temp1
11 rm temp2
12 rm temp3
13 rm temp4
14 rm usage.temp

# format the BBS command use log for output and purge the log file
15 pr /u/bbs/log.file >log.summary
16 > /u/bbs/log.file

# clean out /etc/wtmp
17 > /etc/wtmp

# send both reports to the printer
18 lp usage.summary
19 lp log.summary
```

**End Listings**



FREE OFFER  
of source code  
if purchased before  
Sept 30/87

# AMX 86

KADAK's  
engineers bring  
years of practical real-time  
experience to this mature

## MULTI-TASKING SYSTEM (version 2.0)

for the IBM® PC, PC/XT and PC/AT

- No royalties
- IBM PC DOS® support
- C language support
- Preemptive scheduler
- Time slicing available
- Intertask message passing
- Dynamic operations:
  - task create/delete
  - task priorities
  - memory allocation
- Event Manager
- Semaphore Manager

AMX86™ operates on any 8086/88, 80186/88, 80286 system.

Demo package \$25 US  
Manual only \$75 US  
AMX 86 system \$2195 US  
(shipping/handling extra)

### KADAK Products Ltd.

206-1847 W. Broadway  
Vancouver, B.C., Canada  
V6J 1Y5  
Telephone: (604) 734-2796  
Telex: 04-55670



Also available for 8080, Z80, 68000

CIRCLE 325 ON READER SERVICE CARD

# 80386

## SOFTWARE DEVELOPMENT TOOLS

### The Phar Lap 80386 Software Development Series:

**386|ASM|LINK** by Phar Lap (MS-DOS®) \$495  
Full-featured macro assembler and linker. Includes a debugger and 32-bit protected mode runtime environment.

**80386 High-C™** by MetaWare (MS-DOS®) \$895

**80386 Professional Pascal™** (MS-DOS®) \$895  
by MetaWare

**UNIX™ and VAX/VMS®** cross tools (call)

The wait for professional software development tools for the 80386 is over! Whether you are upgrading an existing IBM PC application to the '386, moving a mainframe application down to a PC, or writing the next PC best-seller, the Phar Lap 80386 Software Development Series is for you. It is an integrated line of products which provides everything you'll need to create and run 80386 protected mode applications under MS-DOS.

(617) 661-1510

Phar Lap Software, Inc. "The 80386 Software Experts"

60 Aberdeen Ave.

Cambridge, MA 02138

CIRCLE 343 ON READER SERVICE CARD

FREE SOURCE CODE!

### Vitamin C Difference

With **Vitamin C**, your applications come alive with windows that explode into view! Data entry windows and menus become a snap. Vitamin C's **open ended design** is full of "hooks" so you can "plug in" special handlers to customize most routines. Of course, Vitamin C **includes all source code FREE**, with no hidden charges. *It always has.*

### Windows

Create windows with one easy function. Vitamin C automatically takes care of complicated tasks like saving and restoring the area under a window.

Options include titles, borders, colors, pop-up, pull-down, zoom-in, scroll bars, sizes to 32k, and more. Unique built-in feature lets users move and resize windows at run-time!

### Data Entry

Flexible dBase-like data entry and display routines feature protected, invisible, required, and scrolling fields, picture clause formatting, full color/attribute control, selection sets, single field and full screen input, and unlimited validation via standard and user definable routines.

# VITAMIN C

It's good for your system!

### High Level Functions

**Standard help handler** provides context sensitive pop-up help messages any time the program awaits key strokes. So easy to use that a single function initializes and services requests by opening a window, locating, formatting, displaying and paging through the message.

**Multi-level MacIntosh & Lotus style menus** make user interfaces and front ends a snap. Menus can call other menus, functions, even data entry screens quickly and easily.

**Text editor windows** can be opened for pop-up note pads and general purpose editing. Features include insert, delete, word wrap, justify, cut, paste, search, and more!

### VCScreen

With VCScreen and Vitamin C working together, you'll reach a new level of productivity you can't reach with a function library alone!

VCScreen speeds development even more! The interactive screen editor actually lets you draw input, output and constant fields, headings, boxes, lines, even a window for your forms to run in.

VCScreen generates readable C source code ready to "plug in" to your application and link with Vitamin C.

FREE SOURCE CODE!

### Guarantee

Better than a brochure. More than a demo disk. If you're not satisfied, simply return the package within 30 days and receive a full refund of the purchase price.

Vitamin C . . . . . \$225.00  
*Includes ready to use libraries, tutorial, reference manual, demo, sample and example programs, and quick reference card; for IBM PC and compatibles. Specify compiler and version when ordering.*

Vitamin C Source . . . . . FREE\*  
\*Free with purchase of Vitamin C.

VCScreen . . . . . \$99.95  
*Requires Vitamin C and IBM PC/XT/AT or true compatible.*

Shipping \$3 ground, \$6 2-day air, \$20 overnight, \$30 overseas. Visa and Master Card accepted. All funds must be U.S. dollars drawn on U.S. bank. Texas residents add 7 1/4% sales tax.

(214) 416-6447

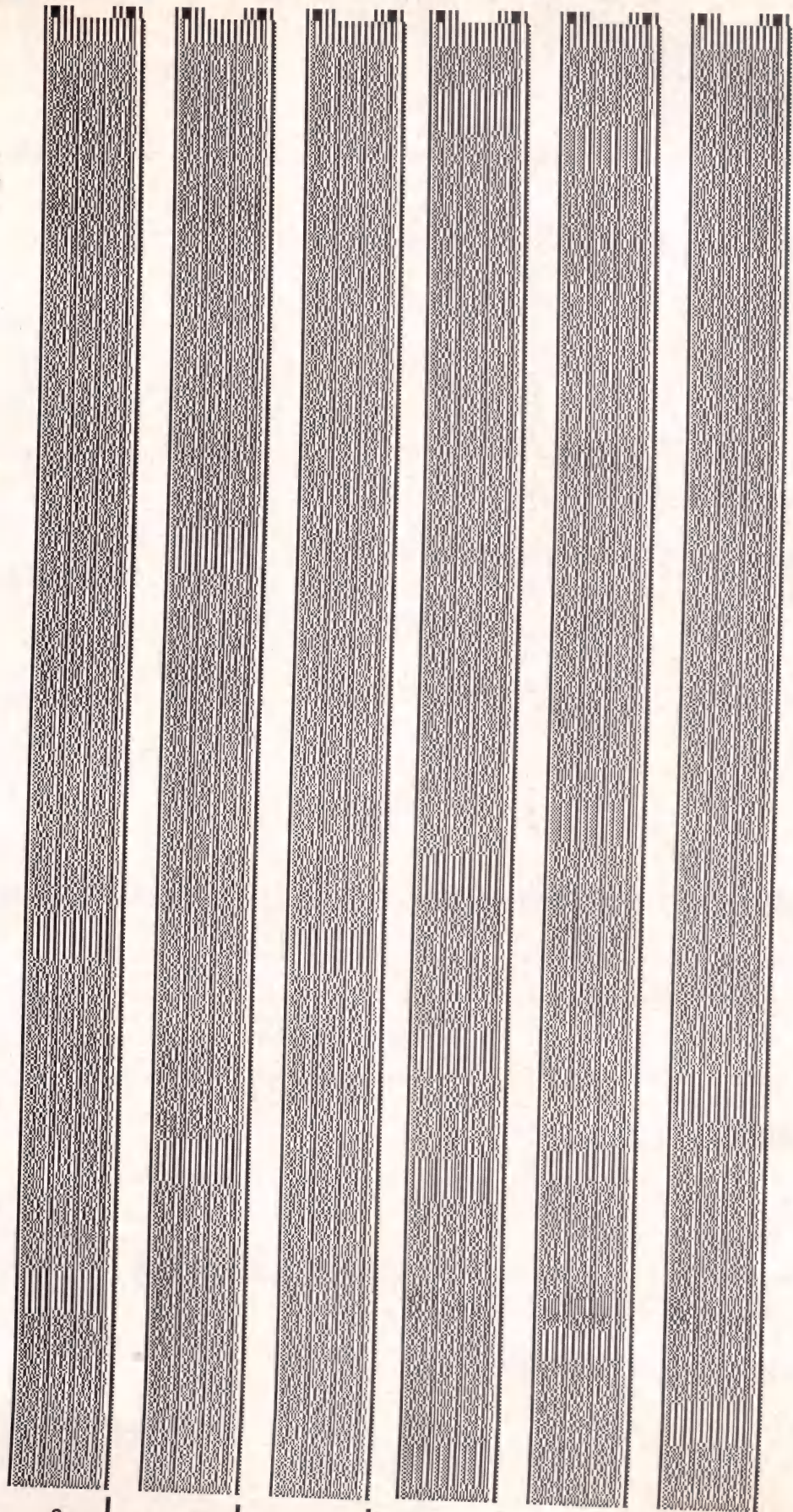
## creative

PROGRAMMING  
Creative Programming Consultants, Inc.  
Box 112097 Carrollton, Texas 75011



# C CHEST

*These softstrips by Cauzin Systems contain the listings for C Chest. The listings begin on page 94.*



1

2

3

4

5

6



## PRODUCTION QUALITY 68020 ANSI C Compiler

If you are doing serious development for the 68020 you already know that existing C compilers do not utilize the processor's full power . . . .

### UNTIL NOW!

- Uses FULL 68020 instruction set
- Full 68881 floating point co-processor support
- Global optimization
- Full ANSI Libraries
- Cuncurrency supported

Available on a variety of hosts  
including VME, VAX, IBM PC & MAXII

**CALL TODAY!**

**817-599-8366**

**Production Languages Corporation**  
P.O. Box 109, Weatherford, Texas 76086

CIRCLE 399 ON READER SERVICE CARD



**The SLR SuperLinker Plus is 3 - 10 times faster than any other linker, and look at these features:**

- link a full 64K output (COM, HEX, SPR or PRL)
- works with Microsoft, Fortran, Basic, Cobol
- supports 32 character externals (SLR format)
- full drive/user support with alternate DU search
- supports 8 address spaces
- fill uninitialized spaces with 0 or FF
- global cross reference
- DSD80/SID compatible .SYM file
- manual overlays
- load map

requires Z80 CP/M 2.2 or greater 32K TPA

**\$195**

**SLR Systems**

1622 N. Main St., Butler, PA 16001  
(800) 833-3061 (412) 282-0864  
Telex 559215 SLR SYS

CIRCLE 78 ON READER SERVICE CARD



## Listing One (Text begins on page 102.)

```

1  #include <stdio.h>
2
3  /* PQ.C      General-purpose priority-queue routines.
4     *          (C) 1987 Allen I. Holub. All Rights Reserved.
5     *
6     * typedef char *PQ;      Dummy typedef for priority queue.
7     *
8     * PQ *pq_create( numele, elesize, cmp, swap, initheap )
9     * int numele;             Max # of elements in the queue
10    * int elesize;            Size of one element in bytes
11    * int (*cmp)();           Pointer to comparison function
12    * int (*swap)();          Pointer to swap function
13    * char *initheap;         Initial heap or NULL to allocate
14    *
15    * pq_ins( p, item )       Insert Item into queue
16    * PQ *p;                  Pointer to priority queue
17    * char *item;             Pointer to item to insert
18    *                          Return number of empty slots
19    *                          before insertion (0 if none).
20    *
21    * int pq_del( p, target ) Delete item from queue
22    * PQ *p;                  Pointer to priority queue
23    * char *target;           Pointer to place to put deleted
24    *                          item.
25    *                          Return # of items in queue before
26    *                          delete (0 if nothing deleted).
27    *
28    * char *pq_look( queue )  Look at (don't delete) top element
29    * PQ *queue;              Pointer to queue.
30    *
31    * int pq_numele( queue )  Return # of elements in queue.
32    * PQ *queue;              Pointer to queue.
33    *
34    *-----*/
35    */
36
37    typedef struct
38    {
39        int (*cmp)(); /* compares two objects */
40        int (*swap)(); /* swap two objects */
41        int itemsize; /* size of one element in heap */
42        int nitems; /* Number of items in the heap */
43        int maxitem; /* Maximum number of items in heap */
44        char *bottom; /* Ptr. to most-recently added item */
45        char *heap; /* pointer to the heap itself */
46    }
47    PQ;
48
49    /*-----*/
50
51    static void reheap_down( p )
52    PQ *p;
53    {
54        /* Reheap the Heap, starting at the top and working
55         * down;
56         */
57
58        int parent; /* index of parent */
59        int child; /* index of child */
60        char *pparent; /* pointer to parent */
61        char *pchild; /* pointer to child */
62        char *psibling; /* pointer to child's sibling */
63        char *heap; /* pointer to heap */
64
65        heap = p->heap;
66
67        for( parent = 0, child = 1; child < p->nitems; )
68        {
69            pparent = heap + (parent * p->itemsize);
70            pchild = heap + (child * p->itemsize);
71
72            if( child+1 < p->nitems )
73            {
74                psibling = pchild + p->itemsize;
75
76                if( (*p->cmp)( pchild, psibling ) < 0 )
77                {
78                    pchild = psibling;
79                    ++child;
80                }
81            }
82
83            if( (*p->cmp)( pparent, pchild ) >= 0 )
84                break;
85
86            (*p->swap)( pparent, pchild );
87
88            parent = child;
89            child = (parent * 2) + 1;
90        }
91    }
92
93    /*-----*/
94
95    static void reheap_up( p )
96    PQ *p;
97    {
98        /* Reheap the Heap, starting at the bottom and working up.
99         * Note that we must use a divide-by-2 rather than a
100         * shift-right in the while loop because -1/2 == 0 but
101         * -1 >> 1 == -1.
102         */
103
104        int parent; /* index of parent */
105        int child; /* index of child */
106        char *pparent; /* pointer to parent */
107        char *pchild; /* pointer to child */
108        char *heap; /* pointer to heap */
109
110        child = p->nitems - 1;
111        heap = p->heap;
112
113        while( (parent = (child-1) / 2) >= 0 )
114        {
115            pchild = heap + (child * p->itemsize);
116            pparent = heap + (parent * p->itemsize);
117
118            if( (*p->cmp)( pparent, pchild ) >= 0 )
119                break;
120
121            (*p->swap)( pparent, pchild );
122            child = parent;
123        }
124    }
125
126    /*-----*/
127
128    PQ *pq_create( numele, elesize, cmp, swap, initheap )
129    {
130        int numele; /* max # of elements in the queue */
131        int elesize; /* size of one element in byte */
132        int (*cmp)(); /* pointer to comparison function */
133        int (*swap)(); /* pointer to swap function */
134        char *initheap; /* initial heap, NULL to allocate */
135
136        /* Create a priority queue that can hold at most
137         * "numele" elements, each of size "elesize". The
138         * cmp function is passed two pointers to queue
139         * elements and it should behave as follows:
140         *
141         * (*cmp)( p1, p2 )
142         *
143         * For descending priority queues [pq_get() returns the
144         * largest item].
145         *
146         * *p1 < *p2 return < 0
147         * *p1 == *p2 return == 0
148         * *p1 > *p2 return > 0
149         *
150         * For ascending priority queues [pq_get() returns the
151         * smallest item].
152         *
153         * *p1 < *p2 return > 0
154         * *p1 == *p2 return == 0
155         * *p1 > *p2 return < 0
156         *
157         * If the initheap argument is NULL, an empty heap is
158         * created automatically, otherwise initheap must point
159         * at an initialized numele-element-long heap.
160         */
161
162        PQ *p, *malloc();
163        int heapsize;
164
165        heapsize = numele * elesize; /* heap size in bytes */
166
167        if( initheap )
168        {
169            if( ! (p = malloc(sizeof(PQ))) )
170                return 0;
171
172            p->heap = initheap;
173            p->bottom = initheap + ((numele - 1) * elesize);
174            p->nitems = numele;
175        }
176        else
177        {
178            if( ! (p = malloc(sizeof(PQ) + heapsize)) )
179                return 0;
180
181            p->heap = (char *) (p + 1);
182            p->nitems = 0;
183            p->bottom = p->heap - elesize;
184
185            p->cmp = cmp;

```



```

185| p->swap = swap;
186| p->itemsize = elesize;
187| p->maxitem = numele;
188| return p;
189| }
190|
191| /*-----*/
192|
193| pq_ins( p, item )
194| PQ *p; /* Pointer to priority queue */
195| char *item; /* Pointer to item to insert */
196| {
197| /* Insert a new item into the priority queue (provided
198| * that space is available.
199| *
200| * Return the number of empty slots in the queue before
201| * the insertion. This number is 0 if the queue is
202| * full and nothing is inserted. Algorithm is:
203| *
204| * if( space is available in the queue )
205| *     increase queue size
206| *     copy new item into bottom of queue
207| *     reheap from the bottom up.
208| */
209|
210| int space_avail = p->maxitem - p->nitems;
211|
212| if( space_avail > 0 )
213| {
214| ++( p->nitems );
215| memcpy( p->bottom += p->itemsize, item, p->itemsize );
216| reheap_up( p );
217| }
218|
219| return space_avail;
220| }
221|
222| /*-----*/
223|
224| int pq_del( p, target )
225| PQ *p; /* pointer to priority queue */
226| char *target; /* place to copy current largest item */
227| {
228| /* Copy the largest item in the priority queue to
229| * the address held in target, then delete the item.
230| *
231| * Return the number of items in the queue before the
232| * delete. If this number is 0, then nothing was
233| * in the queue and *target will not have been
234| * modified. Algorithm is:
235| *
236| * if( there's something in the queue )
237| *     remember pointer to former first item
238| *     replace the first item with the last one
239| *     shrink the heap by one element
240| *     reheap from the top down
241| */
242|
243|
244| int slots_in_use;
245|
246| if( slots_in_use = p->nitems ) /* 1 */
247| {
248|     memcpy( target, p->heap, p->itemsize ); /* 2 */
249|     memcpy( p->heap, p->bottom, p->itemsize ); /* 3 */
250|
251|     --( p->nitems ); /* 4 */
252|     p->bottom -= p->itemsize;
253|
254|     reheap_down( p ); /* 5 */
255| }
256|
257| return slots_in_use;
258| }
259|
260| /*-----*/
261|
262| char *pq_look( queue )
263| PQ *queue;
264| {
265| /* Return a pointer to the largest/smallest element
266| * in the priority queue but don't dequeue it.
267| */
268|
269| return queue->heap;
270| }
271|
272| /*-----*/
273|
274| int pq_numele( queue )
275| PQ *queue;
276| {
277| /* Return number of items in queue
278| */
279|
280| return queue->nitems;
281| }
282|

```

(continued on next page)

## MAMMOTH PROJECTS OFTEN FAIL WITHOUT THE RIGHT TOOLS.



WITHOUT THE PROPER TOOLS, ANY COBOL APPLICATION CAN BE A MAMMOTH PROJECT.

Generate native COBOL screen handling source code for your application.

Or, use COBOL spll's powerful runtime facility.

With COBOL spll, you make the choice! We don't make it for you.

Create interactive demos, tutorials and application prototypes with COBOL spll's dialogue facility.

COBOL spll's powerful panel painter automatically sets attributes, generates automatic borders and lets you move or copy blocks of the panel within or across panels.

Practically all of your field editing can be done with COBOL spll. Perform range and discrete value checking as well as binary value checking.

30 Day Money Back Guarantee!

Only \$345.00! After August 31, 1987 the normal retail price of \$395.00 will go into effect.

Give us a call and we'll send you our free demo. We think you'll be impressed with the power and flexibility of COBOL spll.

COBOL spll supports RM COBOL, Realia COBOL, Microsoft COBOL and RM COBOL 8X.

COBOL spll... THE MISSING LINK TO COBOL PRODUCTIVITY!

Flexus International Corporation  
P.O. Box 9119  
Morristown, NJ 07869  
(201) 895-4724

**flexus**

CIRCLE 189 ON READER SERVICE CARD

## WINDOWS—MENUS—DATA ENTRY—SCREENS

# HI-SCREEN XL™

Application Developers! **HI-SCREEN XL™**, the newest version of HIGH SCREEN, is a complete and unmatched programming tool for managing your user-interface. **HI-SCREEN XL™** features a revamped and enhanced screen editor, increased speed and performance, along with a new thorough reference and user's manual.

Whatever tool you own: compare, you'll be convinced!

### 3. DATA ENTRY

Just define your data entry fields and let **HIGH SCREEN™** do the job: integer/real/alphabetical/date/chosen characters. format/range/upper case-lower case conversion/justify left/right/center/size/color. required input/auto-tabbing/password/Help message. Automatic error handling. Cursor and arrow keys management. Field by field or full screen checking mode for increased flexibility.

- 1. WINDOWS:** for on-line help, menus, data entry. Up to 26 levels deep.
- 2. MENUS:** pop-up, pull-down or Lotus-style, also for batch files.
- 3. DATA ENTRY:** automatic field checking: format, type, range... field by field or full screen. Error handling.
- 4. SCREENS:** full-featured screen editor. Language independent. Fast display from RAM.

Royalty free, not copy protected, 30 day money back guarantee, trade up available.

## Softway, Inc.

PC/SOFT Product Line  
500 Sutter St., Suite 222  
San Francisco, CA 94102

(415) 397-4666  
**HI-SCREEN XL™** is \$149  
(CA res. add tax), S&H USA \$5  
Visa, M/C welcome

**PASCAL, C, dBASE, BASIC, COBOL, FORTRAN, ASSEMBLER, etc.**

CIRCLE 372 ON READER SERVICE CARD



## Listing One (Listing continued, text begins on page 102.)

```

283| /*-----*/
284| #ifdef MAIN
285|
286| int Descending = 1; /* Change these in Codeview */
287| int Makequeue = 1; /* to change the tests. */
288|
289| cmp( s1, s2 )
290| char **s1, **s2;
291| {
292|     int rval;
293|     rval = strcmp( *s1, *s2 );
294|
295|     if( Descending )
296|         return rval;
297|     else
298|         return ( rval < 0 ) ? 1 :
299|                 ( rval > 0 ) ? -1 :
300|                 /* rval == 0 */ 0 ;
301| }
302|
303| /*-----*/
304|
305| swap( s1, s2 )
306| char **s1, **s2;
307| {
308|     char *tmp;
309|
310|     tmp = *s1;
311|     *s1 = *s2;
312|     *s2 = tmp;
313| }
314|
315| /*-----*/
316|
317| printq( p )
318| PQ *p;
319| {
320|     int i;
321|
322|     printf("Queue is:\n");
323|
324|     printf( "\tcmp..... 0x%04x\t", p->cmp );
325|     printf( "\tswap..... 0x%04x\t", p->swap );
326|     printf( "\titemsize..... %d\t", p->itemsize );
327|     printf( "\tnitems..... %d\t", p->nitems );
328|     printf( "\tmaxitem..... %d\t", p->maxitem );
329|     printf( "\tbottom..... 0x%04x\t", p->bottom );
330|     printf( "\theap..... 0x%04x\t", p->heap );
331|     printf( "\tbottom - heap... %d\n", (char **)p->bottom -
332|           (char **)p->heap );
333|
334|     if( p->nitems <= 0 )
335|         printf("\tqueue is empty\n");
336|
337|     else for( i = 0 ; i < p->nitems ; i++ )
338|     {
339|         printf("\t%-2d: %10.10s (0x%04x) (children: %2d, %2d)\n",
340|             i, ((char **)p->heap)[i], ((char **)p->heap)[i],
341|             (2*i)+1, (2*i)+2 );
342|     }
343| }
344| /*-----*/
345|
346| main()
347| {
348|     PQ *queue;
349|     char buf[80];
350|     int i;
351|     char *p;
352|
353|     static char *testq[] =
354|     {
355|         "0", "1", "2", "3", "4", "5", "6", "7", "8", "9"
356|     };
357|
358|     if( Makequeue )
359|         queue = pq_create( 10, sizeof(char*), cmp, swap, 0 );
360|     else
361|         queue = pq_create( 10, sizeof(char*), cmp, swap, testq );
362|
363|     if( !queue )
364|     {
365|         printf("pq_create failed\n");
366|         exit( 1 );
367|     }
368|
369|     printf("-----\n");
370|     printf("Enter i<string><CR> to add string to\n");
371|     printf("queue, d<CR> to dequeue top element, q to\n");
372|     printf("exit the program.\n");
373|     printf("-----\n");
374|
375|     while( 1 )
376|     {

```

```

377|         printq( queue );
378|         printf( "\n[i]d[i]q[<string>] := " );
379|         gets ( buf );
380|
381|         if( *buf == 'q' )
382|             exit( 1 );
383|
384|         else if( *buf == 'i' )
385|         {
386|             i = pq_ins( queue, strsave(buf + 1) );
387|
388|             if( i )
389|                 printf("%d slots avail. before insert\n", i);
390|             else
391|                 printf("Queue was full, did nothing\n\n");
392|         }
393|         else
394|         {
395|             i = pq_del( queue, (char*) &p );
396|             printf("%d slots used before delete, got <ps>\n",
397|                   i, p);
398|
399|             if( i )
400|             {
401|                 free( p );
402|                 p = "nothing";
403|             }
404|         }
405|     }
406| }
407| #endif

```

End Listing One

## Listing Two

Listing 2 -- strsave.c

```

1| char *strsave( str )
2| char *str;
3| {
4|     /* Save the indicated string in a malloc'ed section
5|      * of static memory. Return a pointer to the copy or
6|      * 0 if malloc failed.
7|      */
8|
9|     register char *rptr;
10|     extern char *malloc();
11|
12|     if( rptr = malloc( strlen(str) + 1 ) )
13|     {
14|         strcpy( rptr, str );
15|         return rptr;
16|     }
17|
18|     return (char *)0;
19| }

```

End Listing Two

## Listing Three

Listing 3 -- freq.c

```

1| #include <stdio.h>
2|
3| /* FREQ.C Print a list of the frequency
4|  * of occurrence of all bytes
5|  * in a list of files given on the command line.
6|  * Frequencies are printed as a probability x 100.
7|  * For example, if we read a total of 20 characters,
8|  * 5 of which are 'e', the probability of an 'e'
9|  * occurring in the input is 5/20 (.25) and freq will
10|  * output (.25 * 100) or 25.
11|  */
12|
13| typedef struct
14| {
15|     int val;
16|     double count;
17| } ITEM;
18|
19| #define TABSIZE 256
20| ITEM Tab[ TABSIZE ]; /* I'm counting on this being */
21| /* initialized to zero. */
22|
23| /*-----*/
24|
25| cmp( item1, item2 )
26| ITEM *item1, *item2;
27| {
28|     /* Comparison function used by qsort(), below.
29|      * Count is the primary sort field and val is
30|      * the secondary field.
31|      */
32|
33|     int rval;

```



```

34|
35| return ( item1->count < item2->count ) ? -1 :
36| ( item1->count > item2->count ) ? 1 :
37| /* item1->count == item2->count */
38| item1->val - item2->val ;
39| }
40|
41| /-----*/
42|
43| main( argc, argv )
44| char **argv;
45| {
46|     char *bin_to_ascii(); /* in pchar.c */
47|     FILE *fp;
48|     int i;
49|     double smallest;
50|     double largest;
51|     double numchars = 0.0 ;
52|     double sum = 0.0 ;
53|     double probability = 0.0 ;
54|
55|     reargv( &argc, &argv ); /* Needed only for */
56|                               /* On Command! shell */
57|
58|     for( --argc, ++argv; --argc >= 0; argv++ )
59|     {
60|         if( !(fp = fopen( *argv, "rb" )) )
61|             perror( *argv );
62|         else
63|         {
64|             fprintf( stderr, "%s\n", *argv );
65|
66|             while( (i = getc(fp)) != EOF )
67|             {
68|                 ++ numchars;
69|                 ++ Tab[ i & 0xff ].count ;
70|             }
71|
72|             fclose( fp );
73|         }
74|     }
75|
76|     /* Find largest and smallest elements and at the same
77|      * time initialize the val fields of the Tab entries.
78|      */
79|
80|     largest = 0.0;
81|     smallest = numchars ;
82|
83|     for( i = 0; i <= 0xff; i++ )
84|     {
85|         Tab[i].val = i;
86|
87|         if( Tab[i].count > 0.00000001
88|            && Tab[i].count < smallest )
89|             smallest = Tab[i].count;
90|
91|         if( Tab[i].count > largest )
92|             largest = Tab[i].count;
93|     }
94|
95|     /* Sort the list by probability. A shell sort is used.
96|      * You can replace the ssort call below with a call to
97|      * the qsort() subroutine, available with many compilers.
98|      * Qsort() takes the same arguments as ssort().
99|      * A version of qsort appeared in the in the C Chest,
100|      * DDJ #102 (April, 1985; also Bound Volume 10, p.316).
101|      * The ssort() subroutine appeared in DDJ #113, C Chest
102|      * (March, 1986) p. 70.
103|      */
104|
105|     ssort( Tab, TABSIZE, sizeof(ITEM), cmp );
106|
107|     /*
108|      * Print the list. Each element is printed as three
109|      * numbers: the value of the character (in hex), the
110|      * probability of that character appearing in the input,
111|      * and the probability normalized so that the least-
112|      * frequently occurring probability has the value 1.
113|      */
114|
115|     for( i = 0; i < TABSIZE; i++ )
116|     {
117|         probability = Tab[i].count / numchars ;
118|         sum += probability;
119|
120|         printf( "0x%02x\t%.7f\t%.10f\n",
121|                Tab[i].val,
122|                probability,
123|                Tab[i].count / smallest );
124|     }
125|
126|     fprintf( stderr, "Total = %.6f (N = %.2f)\n",
127|              sum, numchars );
128| }

```

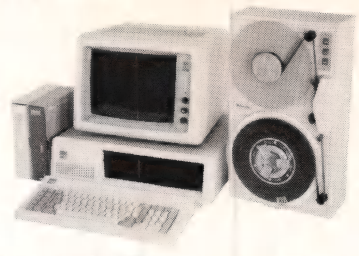
End Listings

## 9-Track Tape Subsystem

for the IBM PC/XT/AT

XENIX or  
MS-DOS.

The solution to your  
micro/mainframe  
communications  
problem is  
available today!



Qualstar's new  
1/2 inch 9-track  
MINISTREAMER™ brings full ANSI data interchange capability  
to the PC. Now you can exchange data files with virtually any  
other computer using 9-track tape.

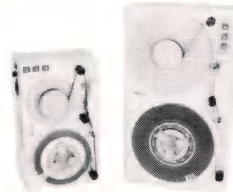
Available in both 7" and 10 1/2" versions, the MINISTREAMER  
weighs in at only 27 pounds and uses less desk space than an  
ordinary sheet of paper, yet provides full 1600/3200 BPI  
capability at an affordable price. Up to 134 megabytes of data  
(depending on format) can be stored on a standard 10 1/2" reel of  
tape, thus making the MINISTREAMER a highly-reliable answer to  
your backup requirements as well.

Tape subsystem includes tape drive, coupler card, cables,  
dust-cover and MS-DOS or XENIX compatible software.  
Prices start at \$2,995.

### 386 READY!

Discover the many advantages  
9-track tape has over other  
Micro/Mainframe links.

Call us today!



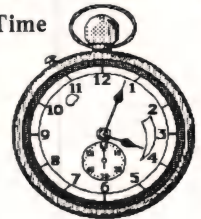
**QUALSTAR®**

9621 Irondale Avenue,  
Chatsworth, CA 91311  
Telephone: (818) 882-5822

CIRCLE 356 ON READER SERVICE CARD

Along With Your Computer, Your Time  
is the Most Important Thing You  
Own. . So Why Waste It?

*Quilt Programmer Productivity  
Tools will help you manage your  
software projects and get control  
of your time!*



#### SRMS™

Software Revision Management System

- Full Featured Revision Control System
- All Versions stored in a Single ASCII File
- Support for Unlimited Libraries
- Support for all programming languages
- Allows you to use your current compilers and editors without conflict
- Windowing Shell interface to simplify use on all library components
- MERGE facility to consolidate different development paths easily, while pointing out conflicting areas
- Full audit trail tracking and reporting on all library components
- Handles big programming projects easily
- Full DOS Pathname and Environment Variable Support
- Requires DOS 2.1+, 224 K, F/H Disk

SRMS Version 3.0.....\$185

#### QMAKE™

Intelligent Program Generation Utility

- Controls the rebuilding of even the most complex systems
- Relieves the developer of remembering which modules need to be rebuilt based on recent changes, how to rebuild them, and in what order to rebuild them
- Works with most compilers, assemblers, and linkers
- Supports full macro definitions, UNIX make-file compatibility, recursive invocations, and command line parameters
- Interfaces completely with SRMS, providing you with a complete set of productivity tools to handle any size project
- Requires DOS 2.1+, 128K F/H Disk

QMAKE Version 1.2.....\$99

SRMS + QMAKE .....\$250

#### NEW ! TXT Tools NEW !

- QSE - Quilt Text Stream Editor
- QSRCH - Quilt File Search Utility (Like UNIX GREP)
- QDIFF - Quilt Windowing File Difference Utility

TXTTOOLS Version 1.0.....\$55

**QUILT  
COMPUTING**

7048 Stratford Road  
Woodbury, MN 55125  
(612) 739-4650



Volume Discounts and Dealer Inquiries Welcome

CIRCLE 107 ON READER SERVICE CARD



**Listing One** (Text begins on page 116.)

Listing 1. Inheritance in SCOOPS

; (C) Copyright 1987 Ernest R. Tello

```

(define-class artifact
  (instvars material weight purpose cost)
  (options
    (gettable-variables material weight purpose cost)
    settable-variables
    inittable-variables))

(define-class transport-means
  (instvars medium time-range power-source)
  (mixins artifact)
  (options
    (gettable-variables medium time-range power-source)
    settable-variables
    inittable-variables))

(define-class transport-vehicle
  (instvars load-capacity length max-speed)
  (mixins artifact transport-means)
  (options
    (gettable-variables load-capacity length max-speed)
    settable-variables
    inittable-variables))

(define-class passenger-vehicle
  (instvars capacity safety dining facilities)
  (mixins artifact transport-means transport-vehicle)
  (options
    (gettable-variables capacity safety dining facilities)
    settable-variables
    inittable-variables))

(define-class water-transport-vehicle
  (classvars (body-name 'hull) (dof 2) (dangers 'sink) (advantages 'relaxing))
  (mixins artifact transport-means transport-vehicle passenger-vehicle)
  (options
    (gettable-variables dof dangers)
    settable-variables
    inittable-variables))

(define-class surface-vessel
  (instvars #-decks #-masts #-engines)
  (mixins artifact transport-means transport-vehicle passenger-vehicle water-transport-
vehicle)
  (options
    (gettable-variables #-decks #-masts #-engines)
    settable-variables
    inittable-variables))

(define-class ship
  (instvars
    x-position y-position x-velocity y-velocity mass)
  (mixins surface-vessel)
  (options
    (gettable-variables x-position y-position x-velocity y-velocity mass)
    settable-variables
    inittable-variables))

(define-method (ship speed) ()
  (sqrt (+ (expt x-velocity 2)
           (expt y-velocity 2))))

(define-method (ship direction) ()
  (atan y-velocity x-velocity))

(define-class ocean-liner
  (instvars company launched homeport tons)
  (mixins ship)
  (options
    (gettable-variables company launched homeport tons)
    settable-variables
    inittable-variables))

(define ship1
  (make-instance ship
    'x-position 100
    'y-position 150
    'x-velocity 30
    'y-velocity 40
    'mass 100))

(compile-class artifact)
(compile-class transport-means)
(compile-class transport-vehicle)
(compile-class passenger-vehicle)
(compile-class water-transport-vehicle)
(compile-class surface-vessel)
(compile-class ship)

```



(compile-class ocean-liner)

## End Listing One

## Listing Two

Listing 2. Multiple Inheritance in SCOOPS

; (C) Copyright 1987 Ernest R. Tello

```
(define-class business
  (instvars name location industry business-type size
    year-founded ownership-type gross-sales costs
    market-share)
  (options
    (gettable-variables name location industry
      business-type size year-founded
      ownership-type)
    (settable-variables gross-sales costs)
    (inittable-variables)))

(define-method (business calc-net-gain) (gross-sales costs)
  (- gross-sales costs))

(define-class adversary
  (instvars aggressiveness allies goals common-goals
    strengths weaknesses)
  (options
    (gettable-variables aggressiveness
      allies goals common-goals
      strengths weaknesses)
    (settable-variables gross-sales costs)
    (inittable-variables)))

(define-class competitor
  (mixins business adversary))

(compile-class business)
(compile-class adversary)
(compile-class competitor)

(define your-business (make-instance business))

(define competitor-1 (make-instance competitor))
```

End Listings

# CANADA'S SOURCE FOR C

- Canadian Sales
- Canadian Service
- Canadian Technical Support
- Canadian Product Knowledge

*We specialize in programming & development software*

LIFEBOAT • LATTICE • GREENLEAF • PHOENIX  
SOFTCRAFT • MICROSOFT • BLAISE • ESSENTIAL  
AGE OF REASON • DESMET • AZTEC  
MARK WILLIAMS • GIMPEL • ROUNDHILL • GSS  
HALO • FAIRCOM • RAIMA • INTEL • etc. • etc. •



Call for full price list—Dealer enquiries welcome



We know our products—we use them!  
**SCANTEL SYSTEMS LTD.**  
801 York Mills Rd., Don Mills, Ont., M3B 1X7  
(416) 449-9252

CIRCLE 391 ON READER SERVICE CARD

## FTL MODULA-2 \$49.95

*The most programming power  
for your money*

- FTL MODULA-2 gives you:**
- Full implementation of MODULA-2.
  - Split screen, multi-file editor.
  - Sources to the run-time modules.
  - Complete support for the product.
  - 8087 reals available for MSDOS.\*

**FTL MODULA-2** meets the standards in Niklaus Wirth's book, "Programming in MODULA-2", third edition. No other language gives you better flexibility, code design, or ease of code maintenance.

**FTL MODULA-2** gives you compact, rommable code quickly. This is the language of the future. If you program in Pascal or C, FTL MODULA-2 will increase your output per programming hour. If you want to learn high level structured language programming, this is the best starting point.

**FTL MODULA-2** was named "product of the year" by Jerry Pournelle in his column, "Computing at Chaos Manor", BYTE Magazine, April, 1986.

\* 8087 support is an additional \$39.95

### FTL Editor toolkit \$39.95

The source code to the editor is offered as a toolkit. The editor was written in FTL MODULA-2 and this set of programs provides the novice with working sources to speed up learning. The experienced programmer will find the wealth of pre-written modules a time-saving bonanza.

### SAVE \$10.00

Buy both the compiler and the editor toolkit for only \$79.95. That's a \$10.00 savings off the regular price. **FTL MODULA-2 is available for both MSDOS and CP/M-80 systems.**



SEND  
CHECK OR  
MONEY ORDER  
TO:

**Workman  
& Associates**

1925 E. Mountain Street  
Pasadena, CA 91104  
(818) 791-7979 voice  
(818) 791-1013 data M-F 8pm-8am  
Or join us on BIX (Byte Information eXchange)  
W.A.N.D.A. Conference.)



MasterCard and Visa welcome.  
Please add \$3.00 Shipping and Handling.  
Calif. residents please add 6% sales tax.

CIRCLE 244 ON READER SERVICE CARD



# YOUR SYSTEM'S KEY COMPONENT

## The Only Magazine By And For Advanced Micro Users.

At last there is a magazine that brings you the strictly technical but practical information you need to stay up-to-date with the ever changing microcomputer technology . . . *Micro/Systems Journal*. *Micro/Systems Journal* is written with the needs of the systems integrator in mind—the individual who's involved in putting together the hardware and software pieces of the microcomputer puzzle.

In each issue of *Micro/Systems Journal* you'll find such useful and progressive articles as:

- Interfacing to Microsoft Windows
- Unix on the PC
- 80386 Programming
- High Resolution PC Graphics
- Using 80286 Protected Mode
- Multiprocessing and Multitasking

You'll get the hands-on, nuts and bolts information, insight and techniques that *Micro/Systems Journal* is famous for . . . in-depth tutorials, reviews, hints . . . the latest information on computer integration, networks and multi-tasking, languages, and operating systems . . . hard-hitting reviews.

To start your subscription to *Micro/Systems Journal*, simply fill out one of the attached cards or write to *Micro/Systems Journal*, 501 Galveston Dr., Redwood City, CA 94063. You'll receive a full year (6 issues) of *Micro/Systems Journal* for just \$20, and enjoy the convenience of having M/SJ delivered to your doorstep each month. Don't wait . . . subscribe today!





# MICRO/ SYSTEMS JOURNAL

FOR THE  
ADVANCED  
COMPUTER  
USER

SUBSCRIBE  
NOW AND

SAVE  
OVER  
15%

OFF THE  
NEWSSTAND  
PRICE!



Yes! I want to subscribe to  
**Micro/Systems Journal™**

and save over 15% off the cover price.

☐ 1 Year (6 issues) \$20 ☐ 2 Years (12 issues) \$35

☐ Please charge my: ☐ Visa ☐ MasterCard ☐ American Express

☐ Payment enclosed ☐ Bill me later

Card # \_\_\_\_\_ Exp. date \_\_\_\_\_

Signature \_\_\_\_\_

Name \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ Zip \_\_\_\_\_

Savings based on a full one-year cover price of \$23.70. Canada and Mexico add \$3 for surface mail, \$7 for airmail per year. All countries add \$12 for airmail per year. All foreign subscriptions must be prepaid in U.S. dollars drawn on a U.S. bank. Please allow 6-8 weeks for delivery of first issue.

A Publication of M & T Publishing, Inc.

3013

\$5  
SAVINGS



Yes! I want to subscribe to  
**Micro/Systems Journal™**

and save over 15% off the cover price.

☐ 1 Year (6 issues) \$20 ☐ 2 Years (12 issues) \$35

☐ Please charge my: ☐ Visa ☐ MasterCard ☐ American Express

☐ Payment enclosed ☐ Bill me later

Card # \_\_\_\_\_ Exp. date \_\_\_\_\_

Signature \_\_\_\_\_

Name \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ Zip \_\_\_\_\_

Savings based on a full one-year cover price of \$23.70. Canada and Mexico add \$3 for surface mail, \$7 for airmail per year. All countries add \$12 for airmail per year. All foreign subscriptions must be prepaid in U.S. dollars drawn on a U.S. bank. Please allow 6-8 weeks for delivery of first issue.

A Publication of M & T Publishing, Inc.

3013

\$5  
SAVINGS



Yes! I want to subscribe to  
**Micro/Systems Journal™**

and save over 15% off the cover price.

☐ 1 Year (6 issues) \$20 ☐ 2 Years (12 issues) \$35

☐ Please charge my: ☐ Visa ☐ MasterCard ☐ American Express

☐ Payment enclosed ☐ Bill me later

Card # \_\_\_\_\_ Exp. date \_\_\_\_\_

Signature \_\_\_\_\_

Name \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ Zip \_\_\_\_\_

Savings based on a full one-year cover price of \$23.70. Canada and Mexico add \$3 for surface mail, \$7 for airmail per year. All countries add \$12 for airmail per year. All foreign subscriptions must be prepaid in U.S. dollars drawn on a U.S. bank. Please allow 6-8 weeks for delivery of first issue.

A Publication of M & T Publishing, Inc.

3013

\$5  
SAVINGS





No Postage  
Necessary  
If Mailed  
In The  
United States

**BUSINESS REPLY MAIL**

FIRST CLASS PERMIT 790, REDWOOD CITY, CA

Postage Will Be Paid By Addressee

*For the Advanced Computer User*

**Micro/Systems Journal**

Box 3713

Escondido, CA 92025-9843



No Postage  
Necessary  
If Mailed  
In The  
United States

**BUSINESS REPLY MAIL**

FIRST CLASS PERMIT 790, REDWOOD CITY, CA

Postage Will Be Paid By Addressee

*For the Advanced Computer User*

**Micro/Systems Journal**

Box 3713

Escondido, CA 92025-9843



No Postage  
Necessary  
If Mailed  
In The  
United States

**BUSINESS REPLY MAIL**

FIRST CLASS PERMIT 790, REDWOOD CITY, CA

Postage Will Be Paid By Addressee

*For the Advanced Computer User*

**Micro/Systems Journal**

Box 3713

Escondido, CA 92025-9843



**MICRO/  
SYSTEMS  
JOURNAL**

**FOR THE  
ADVANCED  
COMPUTER  
USER**

**SUBSCRIBE  
NOW AND**

**SAVE  
OVER  
15%**

**OFF THE  
NEWSSTAND  
PRICE!**



## Publication Quality Scientific Graphics

# Graphic 3.0

color

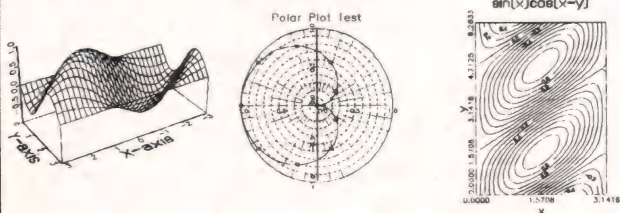
Over 100 C routines make scientific plotting easy

- linear, log, & polar plots
- bar charts & Smith charts
- contour plots with labels
- 3-D curves, 3-D surfaces
- 4 curve types, 8 markers, errorbars
- 14 fonts, font editor
- unlimited levels of superscripts
- 4096 x 3120 resolution in 16 colors on EGA, Tecmar, Sigma boards
- zoom, pan, window and merge plots
- high resolution printer dumps

SOURCE INCLUDED for personal use only

**\$350. Demo \$8**

256k, IBM, AT&T, Corona PCs, DOS 2.xx, 3.xx  
Most boards, printers, and plotters supported  
Microsoft, Latrice, DeSmet, Aztec, CR6 compilers



Scientific Endeavors Corporation

Route 4, Box 79 Kingston, TN 37763 (615) 376-4146

CIRCLE 210 ON READER SERVICE CARD

# 32000

MATH AND FLOATING  
POINT SOFTWARE FOR  
THE NS32000 FAMILY

**32k Math Package: \$750**

Assembly source code for Sin, Cos, Tan, Atan, Log, Exp and Sqrt.  
Single and double precision versions.  
Highly optimized code.

**32k Floating Point Package: \$500**

Assembly source code to emulate the 32081 coprocessor in software.  
Emulation is transparent to user code.  
Allows building of systems with the coprocessor absent or optional.

Price includes right to distribute binary copies in a product without royalties.

Call W.R.I.S.T. Inc. (718) 937-7955  
8-33 40th Ave., L.I.C., N.Y. 11101

CIRCLE 223 ON READER SERVICE CARD

# IQCLISP

More Common Lisp features in less space  
for less money than any other IBM-PC Lisp.

- MSDOS portable
- Bignums, 8087 support
- Multidimensional arrays
- Full Common Lisp package system
- Full set of control primitives.
- Keyword parameters, macros
- Save/restore full environments for speed
- STEP, TRACE, BREAK, DEBUG, ADVISE, APROPOS
- Roll-out frees space for invoking MSDOS commands

**IQCLISP PACKAGE \$300.**

# LISP

Integral Quality

P.O. Box 31970

Seattle, Washington 98103

(206) 527-2918

# IDLISP

Now with a compiler.

- Compiler comes with source
- Compiler cuts execution time 50-75%, program size 60-80%
- Multidimensional arrays
- Floating point, bignums, 8087 support
- Macros
- Color graphics
- Multiple display windows
- Assembly language interface
- Available for IBM PC or TI-PRO

**IDLISP PACKAGE \$270.**  
**INCLUDES COMPILER**

- VISA and Mastercard accepted
- Generous update policy
- Attractive educational license

CIRCLE 327 ON READER SERVICE CARD



## Priority Queues

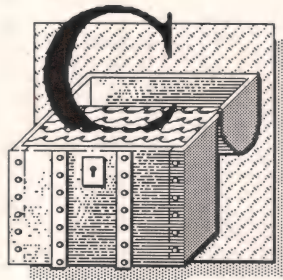
This month is the first of what has unintentionally become a two-part article. I had intended to implement an interesting variant of the Huffman encoding data-compression algorithm that is used by the Unix COMPACT program and is described in Robert Gallager's article "Variations on a Theme of Huffman" (*IEEE Transactions on Information Theory*, vol. IT-24, no. 6 [November 1978]: 668-674). Gallager describes an adaptive one-pass Huffman code in which the code changes as the input file is processed. This way the code tree stays optimal over the life of the file. As usual, the problem proved more intractable than I had at first anticipated. So, this month I'll describe some of the stuff I developed along the way to a solution; I'll discuss the actual compression algorithm in a future column.

### Priority Queues

A *queue*, in the normal data-structure sense of the word, works like a line in a bank does. New entries are added to the back of the queue, and items are removed from the front. That is, the items in the queue are ordered by insertion time—those items

by Allen Holub

that were inserted first are removed first. Another kind of queue is a priority queue or heap, a queuelike data structure in which something other than time is used to order the elements. For example, the largest or smallest element—rather than the least-recently inserted element—could be the first to be dequeued. Heaps have several uses (other than Huffman codes). For example, a heap is used to do the merge phase of the



external-sorting program described in the June 1986 C Chest. A clever sorting algorithm (heapsort) uses a priority queue to do the sorting. Elements are put into the queue in random order, and the smallest element is extracted repetitively until the array is sorted.

The file PQ.C (Listing One, page 94) contains a set of general-purpose priority queue routines. Queues can be constructed of any sort of object (numbers, pointers, structures, and so forth) and can be ordered in any sort of way. You could, for example, keep a queue of structures, adding structures to the queue at random but always extracting the one with the smallest key. You could also create a queue of pointers to structures. You could even use these routines to maintain a normal (but inefficient) queue by adding a time-entered field to the structure and then extracting by smallest time entered. Similarly, a stack can be represented by a priority queue in which the item with the largest time entered is removed first.

PQ.C contains five externally accessible routines. The first of these creates a new queue:

```
typedef char  QUEUE;
QUEUE  *pq_create( numele,
                  elesize, cmp, swap,
                  initheap )
int numele;
int elesize;
```

```
int  (*cmp)( );
int  (*swap)( );
void *initheap;
```

The *QUEUE* type is a dummy pointer-size type that is used in the same way as is the *FILE* pointer returned from *fopen()*. *Pq\_create()* is passed five arguments. *Numele* is the maximum number of elements in the queue. *Elesize* is the size of one element. *Cmp* is a pointer to a comparison function. It is passed pointers to two enqueued objects and should return a value as indicated in Table 1, page 105. *Swap* is a pointer to a swap function. It is passed pointers to two objects, and it should swap the objects (not the pointers). Finally, *initheap* can be used in two ways: if it is *NULL*, an empty priority queue is created; otherwise, *initheap* is assumed to be a pointer to an already-initialized *numele*-long array that will be used as the heap. I'll discuss the utility of this in a moment. I've used the ANSI syntax here to declare *initheap*. A *void* pointer is one that doesn't point at any explicit object; you have to cast it to a pointer to a real type to use it. If your compiler doesn't support *void* pointers, use a pointer to *char*.

A queue that's created by *pq\_create()* can be deleted by *free()*; just pass it the pointer returned from *pq\_create()*. Note, however, that this will only free memory allocated by *pq\_create()* itself. If you create your own initial queue and pass it to *pq\_create()* via the *initqueue* argument, you'll have to free up the memory that you allocated. By the same token, if you create a queue of pointers to strings, *free()* will free the memory used by the queue but not by the strings.

I'll illustrate how to set up a queue



# db\_VISTA<sup>TM</sup>: fast C Database now offers SQL-based Query

## Performance . . .

High-speed data access and low data redundancy . . . just two benefits of using Raima's network model DBMS, db\_VISTA. Combine these benefits with those of C—speed, portability, efficiency and you begin to understand db\_VISTA's real measure . . . performance.

## fast . . .

It's fast — almost 3 times faster than a leading competitor. Fast access that comes from the unique combination of the B-tree indexing method and the "network model" or direct "set" relationships between records. A winning combination for fast performance.

## new db\_QUERY<sup>TM</sup> . . .

Add our new C-linkable, SQL-based, ad hoc query and report-writing companion product to provide a simple relational view of db\_VISTA's complex network model database. Without compromising speed. Ask for db\_QUERY!

## portable . . .

db\_VISTA and db\_QUERY operate on most popular computers and operating systems like MS-DOS, UNIX, and VMS. You can write applications for micros, minis, or even mainframes, allowing the same C applications to run under MS-DOS, UNIX, and VAX VMS.

## multi-user . . .

db\_VISTA has multi-user capability and supports simultaneous users in either multi-tasking or local area networking environments.

## here's how . . .

Design your database and compile your schema file with the database definition language processor. Develop application programs, making calls to db\_VISTA's C functions. Edit and review your database using the Interactive Database Access utility. Compile and link your C program with the db\_VISTA run-time library, and your application is ready to run.

## source code . . .

We make our entire C Source Code available so you can optimize performance or port to new environments yourself.

## efficient . . .

db\_VISTA uses space efficiently. It lets you precisely define relationships to minimize redundant data. It is non-RAM resident; only those functions necessary for operation become part of the run-time program.

## royalty free . . .

Whether you're developing applications for yourself or for thousands, you pay for db\_VISTA or db\_QUERY only once. No extra run-time charges.

## free support . . .

Raima's software includes free telephone support and software updates for 60 days. Technical support personnel are available to answer questions about our software or yours.

## order now . . .

Put db\_VISTA to work in your application program. Ordering is easy — simply call toll-free. We'll answer your technical questions and get you started. Call today.

*Training Classes are available!  
Ask about it!*

BASICS, ADVANCED, INTERNALS  
June 15-19 Seattle, WA USA  
Call for additional dates

	db_VISTA	db_QUERY
■ Single-user	\$ 195	\$ 195
■ Single-user w/Source	\$ 495	\$ 495
■ Multi-user	\$ 495	\$ 495
■ Multi-user w/Source	\$ 990	\$ 990
NEW:		
■ VAX Multi-user	\$ 990	\$ 990
■ VAX Multi-user w/Source	\$1980	\$1980



30 day money-back guarantee!

## db\_VISTA<sup>TM</sup>

### Features

- ◆ Multi-user support allows flexibility to run on local area networks
- ◆ File structure is based on the B-tree indexing method
- ◆ Transaction processing assures multi-user consistency
- ◆ File locking support provides read and write locks
- ◆ SQL-based db\_QUERY is linkable
- ◆ File transfer utilities included for ASCII, dBASE optional
- ◆ Royalty-free run-time distribution
- ◆ Source Code available
- ◆ Data Definition Language for specifying the content and organization of your files
- ◆ Interactive database access utility
- ◆ Database consistency check utility
- ◆ Key file build utility
- ◆ ASCII file import and export utility

### File Management Record and File Sizes

- ◆ Maximum record length limited only by accessible RAM
- ◆ Maximum records per file is 16,777,215
- ◆ Maximum file size limited only by available disk storage
- ◆ Maximum of 256 index and data files
- ◆ Key length maximum 246 bytes
- ◆ No limit on number of key fields per record
- ◆ No limit on maximum number of fields per record

### Operating System & Compiler Support

- ◆ Operating systems: MS-DOS, PC-DOS, UNIX, XENIX, UNOS, ULTRIX, Microport, VMS
- ◆ C compilers: Lattice, Microsoft, IBM, DeSmet, Aztec, Computer Innovations, Turbo C, XENIX and UNIX

## what others say . . .

"If you are looking for a sophisticated C programmer's database, db\_VISTA is it. Raima's customer support and documentation is excellent. Source code availability and a royalty-free run-time is a big plus."

*Dave Schmitt, President  
Lattice, Inc.*

"db\_VISTA has proved to be an all-around high performer in terms of fast execution, flexibility and portability, and has undoubtedly saved us much time and development effort."

*John Adelus, Hewlett-Packard Ltd.  
Office Productivity Division*



# call toll-free today!

**1 (800) db-RAIMA**  
(that's 1-800-327-2462)



**PC/VI™****UNIX's VI Editor Now Available For Your PC!**

Are you being as productive as you can be with your computer? An editor should be a tool, not an obstacle to getting the job done. Increase your productivity today by choosing **PC/VI**—a COMPLETE implementation of UNIX\* VI version 3.9 (as provided with System V Release 2).

**PC/VI** is an implementation of the most powerful and most widely used full-screen editor available under the UNIX operating system. The following is only a hint of the power behind **PC/VI**:

- Global search or search and replace using regular expressions
- Full undo capability
- Deletions, changes and cursor positioning on character, word, line, sentence, paragraph, section or global basis
- Editing of files larger than available memory
- Shell escapes to DOS
- Copying and moving text
- Macros and Word abbreviations
- Auto-indent and Showmatch
- MUCH, MUCH MORE!

Don't take it from us. Here's what some of our customers say: "Just what I was looking for!", "It's great!", "Just like the real VI!", "The documentation is so good I have already learned things about VI that I never knew before." — *IEEE Software*, September 1986.

**PC/VI** is available for IBM-PC's and generic MS-DOS<sup>†</sup> systems for only \$149. Included are CTAGS and SPLIT utilities, TERMCAP function library, and an IBM-PC specific version which enhances performance by as much as TEN FOLD!

**PC/TOOLS™**

What makes UNIX so powerful? Sleek, Fast, and **POWERFUL** utilities! UNIX gives the user not dozens, but hundreds of tools. Now the most powerful and popular of these are available for your PC! Each is a complete implementation of the UNIX program. Open up our toolbox and find:

- |          |         |         |           |
|----------|---------|---------|-----------|
| • BANNER | • DIFFH | • PASTE | • SPLIT   |
| • BFS    | • DIFF3 | • PR    | • STRINGS |
| • CAL    | • GREP  | • RM    | • TAIL    |
| • CHMOD  | • HEAD  | • SED   | • TR      |
| • CUT    | • MAKE  | • SEE   | • TOUCH   |
| • DIFF   | • OD    | • SORT  | • WC      |

All of these for only \$49.00; naturally, extensive documentation is included!

**PC/SPELL™**

Why settle for a spelling checker which can only compare words against its limited dictionary database when **PC/SPELL** is now available? **PC/SPELL** is a complete implementation of the UNIX spelling checker, renowned for its understanding of the rules of English! **PC/SPELL** determines if a word is correctly spelled by not only checking its database, but also by testing such transformations as pluralization and the addition and deletion of prefixes and suffixes. For only \$49.00, **PC/SPELL** is the first and last spelling checker you will ever need!

Buy **PC/VI** and **PC/TOOLS** now and get **PC/SPELL** for only \$1.00! Site licenses are available. Dealer inquiries invited. MA residents add 5% sales tax. AMEX, MC and Visa accepted without surcharge. Thirty day money back guarantee if not satisfied! Available in 5¼", 3½" and 8" disk formats. For more information call today!

\*UNIX is a trademark of AT&T. †MS DOS is a trademark of Microsoft.

**CUSTOM SOFTWARE SYSTEMS**

P.O. BOX 678 • NATICK, MA 01760  
617 • 653 • 2555

**C CHEST**

(continued from page 102)

with a moderately complex example—say you want to keep a queue of pointers to the following structures:

```
typedef struct
{
    int weight;
    char *stuff;
    long more_stuff;
}
ITEM;
```

The *ITEMs* can be inserted in any order, but they will be extracted in order of decreasing weight. The comparison function used for this purpose looks like this:

```
cmp( p1, p2 )
ITEM **p1, **p2;
{
    return (*p1)->weight
        - (*p2)->weight;
}
```

and the swap function looks like this:

```
swap( p1, p2 )
ITEM **p1, **p2;
{
    ITEM *tmp;
    tmp = *p1;
    *p1 = *p2;
    *p2 = tmp;
}
```

If you had wanted to order the queue by increasing—rather than decreasing—weight, you would have reversed *p1* and *p2* in the comparison function's *return* statement.

An empty ten-element queue can now be created with the following:

```
QUEUE *qp;
qp = pq_create( 10, sizeof(ITEM*),
               cmp, swap, 0 );
```

Items are added to the queue using:

```
int pq_ins( qp, item )
QUEUE *qp;
char *item;
```

where *qp* is a pointer returned from a previous *pq\_create( )* call and *item* is a pointer to the object to insert. *Pq\_ins( )* returns the number of empty slots that were in the queue



before the insert. If the return value is 0, the queue was full and `pq_ins()` will have done nothing. A new item is inserted in the queue that I created earlier with the following code:

```
ITEM *p;
p = (ITEM *) malloc(sizeof(ITEM));
p->stuff = "A string";
p->weight = 5;
if( !pq_ins( qp, &p ) )
    printf("queue is full");
```

Note that you have to pass in the address of the object to enqueue—in this case the address of the pointer `p`. Items are extracted from the queue with:

```
int pq_del( qp, item )
QUEUE *qp;
char *item;
```

where, again, `qp` is a `QUEUE` pointer returned from `pq_create()` and `item` is a pointer to a place into which the dequeued object will be copied. The number of items in the queue before the delete is returned. If this number is 0, the queue was empty and the contents of `*item` are undefined. For example, the largest element of the queue can be dequeued with the following code:

```
ITEM *p;
if( !pq_del( qp, &p ) )
    printf("queue is empty");
```

Two additional support routines are provided:

```
char *pq_look( p )
char *pq_numele( p )
QUEUE *p;
```

Again, `p` is a pointer to a `QUEUE` returned from a previous `pq_create()` call. `Pq_look()` returns a pointer to the object at the head of the queue. The object is not actually dequeued, however. `Pq_numele()` returns the number of elements currently in the queue.

### Implementation

A priority queue can be represented by a binary tree that has the following properties:

1. All children in the tree have a value less than their parent.

2. The tree is as perfectly balanced as possible—that is, the difference in height of all leaves is at most 1.

3. Leaves are inserted into the tree from left to right until an entire rank is full, then the next rank is started.

An example tree is shown in Figure 1, below. Note that this tree is not strictly ordered in the normal way. That is, rule 1 doesn't require that the tree be sorted, only that the children are smaller than the parent. You could exchange the subtrees rooted at 1 and 2 without violating rule 1. This ordering guarantees that the root node of the entire tree always holds the largest element, however. Notice that an insert operation is harder than normal because of rules 2 and 3. An 11th node must be inserted as the right child of node 4 and a 12th node as the left child of node 5. If you do this, however, rule 1 may be violated by the newly inserted node. To avoid this last problem, you have to adjust the contents of the nodes at every insertion (a process

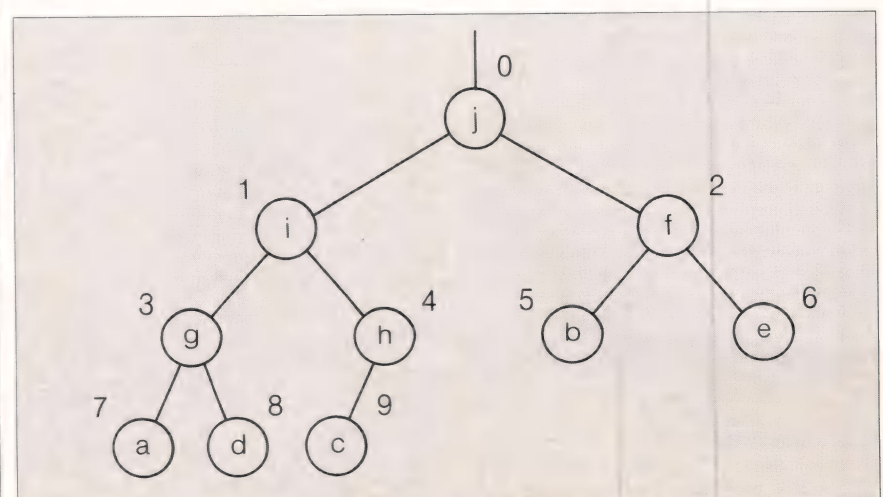
called reheap). For example, if you insert a new node having the value `k` as the right child of node 4, you'll have to shuffle things around because of rule 1.

The reheap process is illustrated in Figure 2, page 106. You reheap from the bottom up with an insertion. Because `k` is greater than `h`, the contents of nodes 10 and 4 must be swapped. You then go up one level to node 4 and compare its key to its parent's. Here `k` is still greater than `i`, so you swap again, moving the `k` to node 1. Finally, nodes 0 and 1 must be swapped as well, moving `k` to the root position.

Because of rules 2 and 3, it's convenient to represent the tree as an array in which the element at node `N` has children at nodes `2N+1` and `2N+2`. For example, the node at `array[0]` has children at `array[1]` and `array[2]`, the node at `array[2]` has children at `array[5]` and `array[6]`, and so forth. This array representation, usually called a heap, has both advantages and disadvantages. The main

For an ascending queue (the smallest object is dequeued first):		For a descending queue (the largest object is dequeued first):	
if:	(*cmp)(p1, p2) returns:	if:	(*cmp)(p1, p2) returns:
<hr/>			
*p1 > *p2	negative number	*p1 > *p2	positive number
*p1 == *p2	0	*p1 == *p2	0
*p1 < *p2	positive number	*p1 < *p2	negative number

**Table 1:** Values returned from the comparison function



**Figure 1:** A priority queue represented as a binary tree



problems are that the maximum number of elements must be known at create time, and it's difficult to delete an interior node at random or to merge two queues. Nonetheless, the heap representation is the most efficient for the overwhelming majority of applications—in which the only operations are insert and delete.

The tree from Figure 1 is shown as an array in Figure 3, right. To insert a node in the tree, you make the array one element larger, put the new node in the rightmost element, and then reheap from the bottom up (right to left). The largest node is always at *array[0]*, just as it was always at the root position in the tree. Delete the largest element by copying the rightmost node in the tree (*array[9]* in Figure 3) into *array[0]*, making the array one element smaller, and then reheap from the top down (left to right).

The heap representation is a reasonably efficient one. Unlike a normal binary tree, you never have to search for the insertion point in the tree (it's always at the far right). Similarly, the largest element is always in a fixed place (at the far left). Though there's a certain amount of copying that has to be done during a reheap, no more than  $\log_2 N$  copies need ever be done. Nonetheless, it's worthwhile to make the queue elements themselves small. Use an array of pointers to structures rather than an array of structures. This way you have to swap only two pointers—rather than two complete structures—when you reheap.

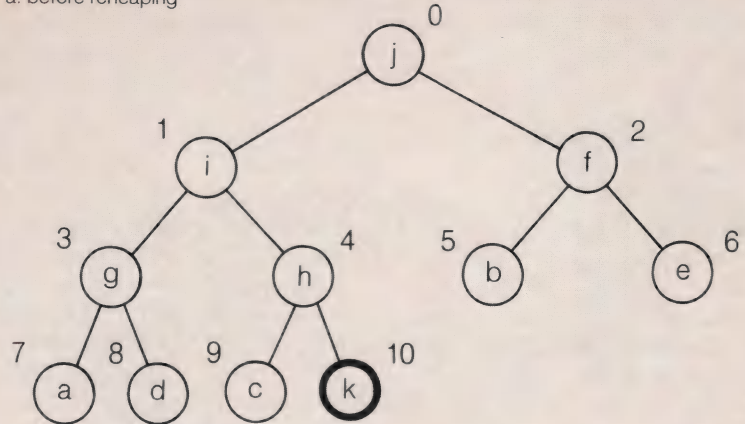
Note that a sorted array is a valid heap (though a heap is not necessarily a sorted array). This property explains the *initheap* argument to *pq\_create()*. If your input is already a sorted list, there's no point in doing a series of insert operations to create the heap—you can just pass the array directly to *pq\_create()*.

The priority queue is represented internally by the following structure:

```
typedef struct
{
    int (*cmp)();
    int (*swap)();

```

a. before reheap



b. after reheap

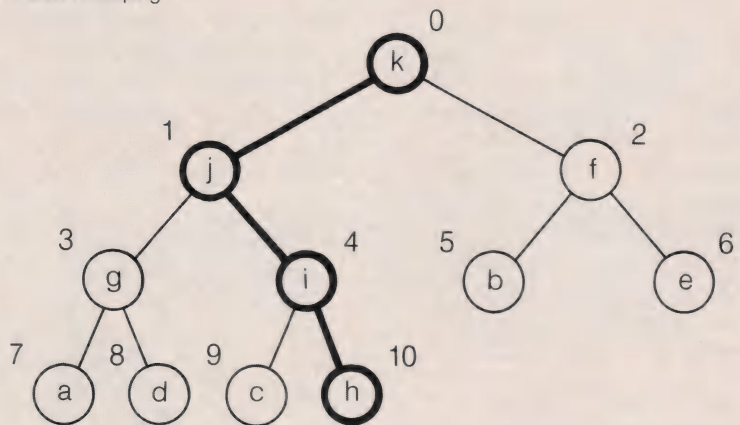


Figure 2: Inserting a new node

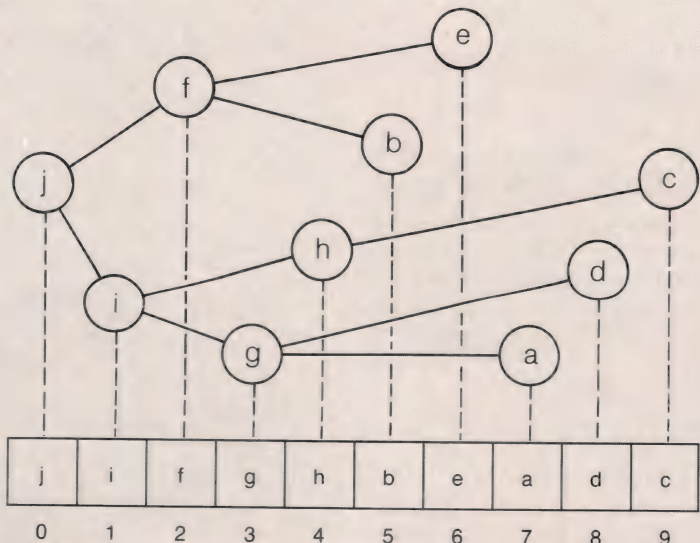


Figure 3: A priority queue represented as an array



# C CODE FOR THE PC

*source code, of course*

## C Source Code

FSP (screen manager)	\$400
GraphiC 3.0 (high-resolution, DISSPLA-style scientific plots in color & hardcopy)	\$300
Essential C Utility Library (400 useful C functions)	\$160
Essential Communications Library (C functions for RS-232-based communication systems)	\$160
Panache C Program Generator (screen-based database management programs)	\$150
B-Tree Library & ISAM Driver (file system utilities by Softfocus)	\$100
The Profiler (program execution profile tool)	\$100
QC88 C compiler (ASM output, small model, no longs, floats or bit fields, 80+ function library)	\$90
CBTree (B+tree ISAM driver, multiple variable-length keys)	\$80
ME (programmer's editor with C-like macro language)	\$75
Wendin PCNX Operating System Shell	\$75
Wendin PCVMS Operating System Shell	\$75
Wendin Operating System Construction Kit	\$75
Entelekon C Function Library (screen, graphics, keyboard, string, printer, etc.)	\$70
Entelekon Power Windows (menus, overlays, messages, alarms, file handling, etc.)	\$70
EZ_ASM (assembly language macros bridging C and MASM)	\$60
Make (macros, all languages, built-in rules)	\$50
Coder's Prolog (inference engine for use with C programs)	\$45
PC/MPX (light-weight process manager; includes preemption and coroutine packages)	\$45
Biggerstaff's System Tools (multi-tasking window manager kit)	\$40
TELE Kernel (Ken Berry's multi-tasking kernel)	\$30
TELE Windows (Ken Berry's window package)	\$30
Clisp (Lisp interpreter with extensive internals documentation)	\$30
Translate Rules to C (YACC-like function generator for rule-based systems)	\$30
6-Pack of Editors (six public domain editors for use, study & hacking)	\$30
ICON (string and list processing language, Version 6 and update)	\$25
LEX (lexical analyzer generator)	\$25
Bison & PREP (YACC workalike parser generator & attribute grammar preprocessor)	\$25
C Compiler Torture Test (checks a C compiler against K & R)	\$20
A68 (68000 cross-assembler)	\$20
Small-C (C subset compiler for 8080 and 8088)	\$20
tiny-c (C subset interpreter including the tiny-c shell)	\$20
Xlisp 1.5a (Lisp interpreter including tiny-Prolog in Lisp)	\$20
List-Pac (C functions for lists, stacks, and queues)	\$20
XLT Macro Processor (general purpose text translator)	\$15
C Tools (exception macros, wc, pp, roff, grep, printf, hash, declare, banner, Pascal-to-C)	\$15

## Data

Webster's Second Dictionary (234,932 words)	\$60
U. S. Atlas (29,000 cities with retrieval program)	\$40
KST Fonts (13,200 characters in 139 mixed fonts: specify T <sub>E</sub> X or bitmap format)	\$30
The World Digitized (100,000 longitude/latitude points)	\$30
NBS Hershey Fonts (1,377 stroke characters in 14 fonts)	\$15
U. S. Map (15,701 points)	\$15

The Austin Code Works  
11100 Leafwood Lane  
Austin, Texas 78750-3409  
(512) 258-0785

Free shipping on prepaid orders

CIRCLE 250 ON READER SERVICE CARD

MasterCard/VISA



```
int itemsize;
int nitems;
int maxitem;
char *bottom;
char *heap;
```

```
{
PQ;
```

The comparison and swap function pointers are remembered in *cmp* and *swap*. *Itemsize* is the size in bytes of one element of the heap, *nitems* is the number of elements currently in the heap, and *maxitem* is the maximum number of items that the heap can hold. The *heap* field points at the array itself, and *bottom* points at the

most recently added element. It will point at *heap[-1]* if the heap is empty. Note that you could derive *nitems* by subtracting *bottom* from *heap* but it's more convenient to keep it as a separate number.

The *pq\_create()* function on lines 128–188 of Listing One creates a *PQ* structure and initializes the various fields. If *initheap* is not *NULL*, space for the *PQ* structure alone is allocated (on line 167) and *heap* is made to point to the specified array. The various other fields are adjusted so that the heap is full. If *initheap* is *NULL*, space for both the *PQ* structure and the heap itself is allocated (on line 176), and an empty heap is initialized. Finally, a pointer to the *PQ* structure is returned on line 188.

The insert and delete operations are performed in *pq\_ins()* (lines 193–219) and *pq\_del()* (lines 224–258) using the method described earlier. Because the compiler doesn't know the size of a heap element at compile time, items must be copied with *memcpy()* calls at run time. The reheaping is deferred to two static workhorse functions—*reheap\_down()* and *reheap\_up()*—declared at the top of the file. *Reheap\_down()* (lines 51–89) starts at the root node (*heap[0]*) and works down the tree. It selects the larger of the two children on lines 72–81 and then swaps the root element and the larger child (and moves down the tree) if necessary. The reheap process stops as soon as the root is larger than both

## Flotsam and Jetsam

### Standard #include Files

At present I'm using the Microsoft C compiler, Version 4.0, for two reasons—the first is the CodeView debugger, and the second is the degree of Unix compatibility. (The MS-DOS compiler is the Xenix compiler; there's literally no difference.) I spend a lot of time bouncing around between the Unix system at Berkeley and my own PC, so having compatible compilers is essential. Using the Microsoft compiler, I've never had to modify a program written under one system and ported to the other unless that program did some sort of very-low-level communication with the operating system (such as talk directly to the BIOS).

There's another advantage to Unix compatibility, and that's standardization. Although the emerging ANSI standard finally incorporates the I/O library into the language—at least in terms of regularizing the function names and argument-passing conventions—there aren't nearly as many ANSI functions as there are Unix functions, so Unix must continue to provide a de facto standard. You don't really know C unless you know the Unix library. Your code just won't be portable because you don't know what's standard and what isn't. Learning your own compiler's library is a necessary but not

sufficient condition for knowing C. As a consequence, it's very much to your advantage to pick up a copy of the Unix System V documentation and read through it. Hitherto, these manuals have been hard to come by. All that you could get were the now-out-of-date Version 7 manuals (the big blue and green paperbacks). AT&T, however, has just published the manuals for Release 2.0 of Unix System V. The complete five-volume set is overkill unless you're really using Unix. Nonetheless, Volume 2—*System Calls and Library Routines*—should be in every C programmer's library. It's available by mail order (for \$29.95) from the Computer Literacy bookshop in San Jose, California, and is well worth the money. (Computer Literacy's phone number is [408] 435-1118. The five-volume set is Steven V. Earhart's *Unix Programmer's Manual* [New York: Holt, Rinehart & Winston, 1986].)

Because I develop all the programs that appear in C Chest in the Unix environment (the Microsoft environment is the Unix environment for all intents and purposes), it seems worthwhile to list the various *#include* files that I use in C Chest. These are usually presented without comment because they're both standard and well documented elsewhere. If your compiler doesn't have one of

these files, then you don't need to *#include* it in your program because none of the library functions will require any of the things included within the file. By the same token, if your compiler doesn't have one of these files, then it's not Unix compatible, in spite of what the advertisements may say.

It's impossible for me to describe how to port programs to every non-standard compiler on the market. You'll have to learn how to do this yourself. You'll need to know your own compiler's library pretty well and have at least a working knowledge of the Unix equivalents to various library functions. Do your homework. I will say, while on the subject, that the Lattice compiler is one of those that falsely claims Unix compatibility—for example, it doesn't support a Unix-compatible *stat()* function, it uses the name *fork()* in incorrect ways, and it doesn't have a Unix-compatible *#include* file system. These inconsistencies are surprising because Lattice seems to have gone to a lot of trouble to add Unix-compatible functions to its library in the last release. Unfortunately, the degree of Unix compatibility that is indeed present lures you into a false sense of security.

The various *#include* files that you're likely to see in a C Chest pro-



children. Note that an ascending queue (one where the root holds the smallest rather than the largest element) can be created by modifying the comparison function, without touching the reheap code at all.

*Reheap\_up()* (lines 95–124) reheaps in the other direction. It starts with the most recently entered item (which is at *heap[nitems-1]*) and works up the tree to the root. The routine is simpler than *reheap\_down()* because you don't have to find the larger sibling—a child has only one parent. Again, the reheaping processes stops as soon as a parent node that is larger than the current child node is detected.

The remainder of the file is compiled only if *MAIN* is *#defined*. In this

case, a stand-alone test program is compiled. This program creates a ten-element-long heap of character pointers and then adds or deletes strings from the heap according to commands entered from the keyboard. Type *i*<*string*> to insert <*string*> into the heap, *d* to delete an item, and *q* to exit from the program. The priority queue is created either on lines 359 or 361, depending on the value of *Makequeue*. If *Makequeue* is set, an empty queue is created; otherwise, a preinitialized queue (using the array declared on lines 353–356) is created. If *Ascending* is true, the queue is an ascending priority queue (items will be removed in ascending order); otherwise, it's descending. There's no explicit code to

modify *Makequeue* and *Ascending* (I just modified them with CodeView, the Microsoft debugger, as I was debugging).

New items are inserted into the queue on line 386 and deleted on line 395. The comparison and swap functions are declared on lines 289–313. Finally, the *strsave()* function, used on line 389, is shown in Listing Two, page 96. Because this is just a small test program, I'm ignoring the error return from *strsave()*—you shouldn't do this in a real application, of course.

### **FREQ.C**

The priority queue routines are of general utility. You need a few special-purpose utilities to make Huff-

gram are listed below. Most of these are both Unix and Microsoft compatible, but some are used just in the DOS compiler. Note that this isn't a list of all the Unix/Microsoft *#include* files—just the ones I'm likely to use.

*ctype.h*—contains various text processing macros: *isalpha*, *isupper*, *islower*, *isdigit*, *isxdigit*, *isspace*, *ispunct*, *isalnum*, *isprint*, *isgraph*, *isctrl*, *isascii*, *toupper*, *tolower*, and *tascii*.

*dos.h*—not Unix compatible. Contains *#defines* for the MS-DOS interface functions: *bdos*, *dosexterr*, *int86*, *int86x*, *intdos*, and *segread*.

*errno.h*—*#defines* for the various error condition codes returned from the I/O library: *EZERO*, *EPERM*, *ENOENT*, *ESRCH*, *EINTR*, *EIO*, *ENXIO*, *E2BIG*, *ENOEXEC*, *EBADF*, *ECHILD*, *EAGAIN*, *ENOMEM*, *EACCES*, *EFAULT*, *ENOTBLK*, *EBUSY*, *EEXIST*, *EXDEV*, *ENODEV*, *ENOTDIR*, *EISDIR*, *EINVAL*, *ENFILE*, *EMFILE*, *ENOTTY*, *ETXTBSY*, *EFBIG*, *ENOSPC*, *ESPIPE*, *EROFS*, *EMLINK*, *EPIPE*, *EDOM*, *ERANGE*, *EUCLEAN*, and *EDEADLOCK*.

*fcntl.h*—contains definitions needed to use the unbuffered I/O function *open()*: *O\_RDONLY*, *O\_WRONLY*, *O\_RDWR*, *O\_APPEND*, *O\_CREAT*,

*O\_TRUNC*, *O\_EXCL*, *O\_TEXT*, and *O\_BINARY*.

*io.h*—contains function declarations for *access*, *chmod*, *chsize*, *close*, *creat*, *dup*, *dup2*, *eof*, *filelength*, *isatty*, *locking*, *lseek*, *mktemp*, *open*, *read*, *rename*, *setmode*, *sopen*, *tell*, *umask*, *unlink*, and *write*.

*math.h*—contains definitions for *abs*, *acos*, *asin*, *atan*, *atan2*, *atof*, *bessel*, *cabs*, *ceil*, *cos*, *cosh*, *exp*, *fabs*, *floor*, *fmod*, *frexp*, *hypot*, *labs*, *ldexp*, *log*, *log10*, *matherr*, *modf*, *pow*, *sin*, *sinh*, *sqrt*, *tan*, and *tanh*.

*process.h*—contains definitions for various process-control functions: *abort*, *execl*, *execle*, *execlp*, *execlepe*, *execv*, *execve*, *execvp*, *execvpe*, *exit*, *\_exit*, *getpid*, *spawnl*, *spawnle*, *spawnlp*, *spawnlpe*, *spawnv*, *spawnve*, *spawnvp*, *spawnvpe*, and *system*.

*signal.h*—contains definitions needed by the *signal()* subroutine. Some common definitions are: *SIGINT*, *SIGFPE*, *SIG\_DFL*, and *SIG\_IGN*.

*stdarg.h*—contains definitions needed to write an ANSI-compatible subroutine with a variable number of arguments (see *varargs.h*). The following are defined: *va\_list*, *va\_start*,

*va\_arg*, and *va\_end*.

*stdio.h*—contains *#defines* and so on for all the buffered I/O functions (*fopen*, *fprintf*, and so forth). *Stdin*, *stdout*, *stderr*, *FILE*, *EOF*, and *NULL* are all defined here. Note that several pseudofunctions that you're used to thinking of as subroutines (*getchar*, *putchar*, *getc*, *putc*, *feof*, *ferror*, and *fileno*) are actually macros that are *#defined* in *stdio.h*. If you forget to *#include* this file, you'll get error messages from the linker (such as "Unresolved external: \_putchar"). *Putchar* is a macro that ultimately evaluates to a call to a system-level subroutine usually called either *\_flsbuf* or *\_flushbuf*.

*sys/stat.h*—contains definitions for the system-status subroutines *stat()* and *fstat()*.

*sys/types.h*—also required by *stat()* and *fstat()*. Various time functions (such as *utime()*) use the information declared here as well.

*varargs.h*—Definitions for the variable-number-of-argument mechanism used by Unix System V (see *stdarg.h*). The following are defined here: *va\_list*, *va\_dcl*, *va\_list*, *va\_arg*, *va\_start*, *va\_arg*, and *va\_end*. ☐



man trees, however. One of these is the `FREQ.C` program shown in Listing Three, page 96. `FREQ.EXE` is a standalone program that takes as input a list of files and outputs a table showing the frequency of occurrence of every 8-bit pattern in these files (I'll call these patterns "characters" from here on). The output table is sorted in ascending order of frequency (least frequently occurring characters toward the top). The output format uses one character per line with three numbers on each line—the leftmost number is the value of the character itself (in hex); the next two numbers are character frequencies, output in two forms. The first form is the probability of occurrence of every character—the number of times that that character occurred in the input divided by the total number of input characters. The second form is a normalized character count in which the least frequently occurring nonzero pattern has the value of 1—it's the character count divided by the count associated with the least frequently occurring character.

The table itself is declared on lines 13–21. It is a 256-element array of `ITEMs`, indexed by character value. One field of the `ITEM` structure holds a count that is incremented every time the associated character is encountered in the input. The other

field holds the character value itself. You can derive this from the index, of course, but putting it into an `ITEM` lets you sort the array by frequency of occurrence without losing the value.

The array is loaded with the `for` loop on lines 58–74. The count associated with every byte is incremented on line 69. The counts are all initialized to 0 by default because global-level objects are always initialized to 0 unless there's an explicit initializer present as part of the declaration. The next `for` loop, on lines 83–93, initializes the `value` fields of the array and at the same time finds the element having the smallest nonzero value. (I'm getting the largest value, too, but am not using it for anything at present.) The array is sorted with the `ssort()` call on line 105. This subroutine works just like the Unix-compatible `qsort()` does, but it does a shell sort. See the comment on lines 95–102 for more information. The comparison function used by `ssort()` is declared on lines 25–39. Finally, the table is printed (and the probabilities and so forth are computed) in the loop on lines 115–126. The sum of the probabilities is printed to `stderr` on line 126 just to make sure that everything worked correctly. It should always be 1.00.

#### Availability

All the source code for articles in this issue is available on a single disk. To

order, send \$14.95 to *Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94063 or call (415) 366-3600 ext. 216. Please specify the issue number and format (MS-DOS, Macintosh, Kaypro).

#### Bibliography

Fredman, Sedgewick, Sleator, and Tarjan. "The Pairing Heap: A New Form of Self-Adjusting Heap." *Algorithmica* 1:1 New York: Springer Verlag, 1986: 111–129. This describes a priority queue represented as a tree (in order to make the merge and random-delete operations more efficient).

Holub, Allen. "C Chest." *DDJ* (June 1986): 26–40. This article describes a set of general-purpose queue-manipulation routines. The same article appears in *DDJ Bound Volume 10*: 436–334.

Sedgewick, Robert. *Algorithms*. Reading, Mass.: Addison-Wesley, 1983: 127–141. This book contains a description of priority queues (or heaps) and heapsort.

DDJ

(Softstrips begin on page 92.)

(Listings begin on page 94.)

Vote for your favorite feature/article.  
Circle Reader Service No. 8.

## Introducing Periscope™ III

**A new generation of debugging for the IBM PC, XT, AT and close compatibles**

Now you can invest \$995 and get the most powerful debugging tool available short of a \$10,000 in-circuit emulator! The Periscope III board's hardware breakpoints and real-time trace buffer help you solve the really tough debugging problems. If you ever deal with errors in real-time systems, intermittent failures, interfacing with undocumented systems, or bottlenecks in your code, Periscope III may be just what you need!

**Call TOLL-FREE 800/722-7006 for more information.**

The  
**PERISCOPE**  
Company, Inc.

14 Bonnie Lane • Atlanta, GA 30328 • 404/256-3860

CIRCLE 214 ON READER SERVICE CARD



# Ever Program On A Silver Platter??

How much would you expect to pay for a 32 bit MC 68000 computer that's a mainframe condensed down into a keyboard? How about \$389.00!!!!? If it makes you feel any better simply add a zero to the price when you order! But that's actually our price!!! The most powerful computer money can ever buy is now the most inexpensive computer money can buy!!! So don't buy the name! Buy the power!! The power is **not** in the name!

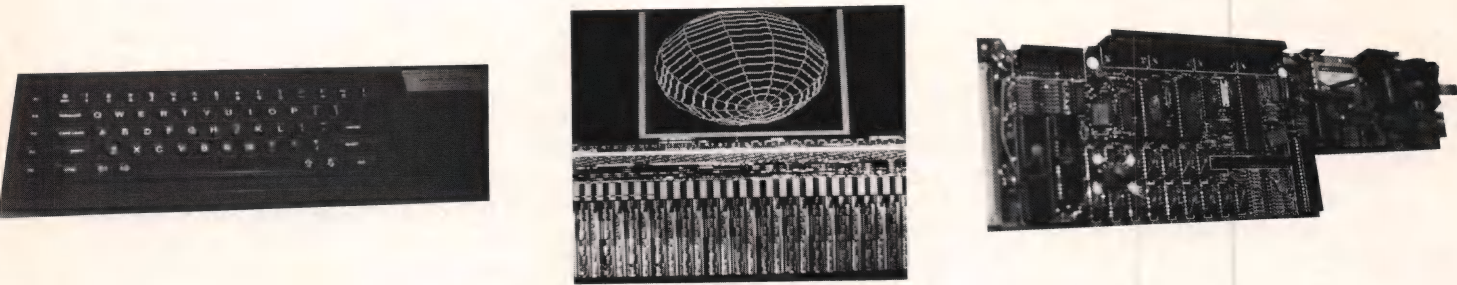
If **you** had the opportunity to work amongst Machine Code ROM Designers, VAX & UNIX wizards in a research laboratory, designing an MC 68000 based computer that's 2nd to none. . .

What would you come up with?? And what would you call it??

**Well It's Already Been Done!!**

**They Called It The QL For The Quantum Leap It Is!!**

**Absolutely a Quantum Leap beyond what you know & use - and it's truly like Programming on a Silver Platter!!**



**The QL Desktop Minicomputer:** Designed by SRL Labs, manufactured by Samsung. An absolute Quantum Leap beyond all the rest! The phenomenal open architecture QDOS: with Virtual Memory RAM, Multitasking Job Control, Multiuser Networking. It'll Cache Files into unused Memory and create/delete Directories Automatically! Even allows File Names up to 36 characters long! Everything is built into ROM here: QDOS, Networking, Windowing, & 32 Bit SuperBasic, all in a totally concurrent non-destructive environment. Unlimited quantities & lengths allowed with: Variables, Program Lines, CONsoles & Buffers. Dynamic non-destructive virtual RAM Disking & Networking buffers too! Even a System Variables Brain Page Screen! Built-in DCE & DTE Serial Ports.

#### Language Environments:

Metacomco's "C", LISP, BCPL, 68000 Assembler, APL, Development Kits. Prospero's Pro Pascal & Pro Fortran 77. Digital Precision's Forth-83. QJUMP's 65C02 or 8088 Cross Assembly ROMs. Everything generates native 68000 Compiled Code. ROM Firmware & Software Package is now available which will even bring it up in CPM!

Imagine working with a 32 bit SuperBasic that's structured like Turbo Pascal, powered beyond PIC Basic, in an interpreter always present with QDOS, all concurrently running in a built-in UNIX-like multitasking job controlled environment with access to 360 fully channeled windows, devices & files by EACH job! 3 Major Compilers already exist for the SuperBasic source alone! TURBO, SUPERCHARGE, QLBERATOR! The compiled SuperBasic code or ANY other language will multitask and control with QDOS and SuperBasic. The list of ALL the Superior Features would fill this entire publication!

The QL comes bundled WITH PSION Integrated Word Processor, Spreadsheet, Database and Presentation Graphics Programs. PLUS: Our FREEWARE Demos & Utilities with all purchases! Plus \$12 Shipping and Handling

**Call: (201) 328-8846**

QLine BBS: 328-2919



Technical Info & Assistance - -

Telex: 9102500026

Compuserve ID # 76625,2214

**Quantum Computing, Box 1280, Dover, NJ 07801**



CIRCLE 144 ON READER SERVICE CARD



## Resources

Ward, Robert. *Debugging C*. Indianapolis, Ind.: Que Corp., 1986. 350 pages with index. ISBN 0-88022-261-1.

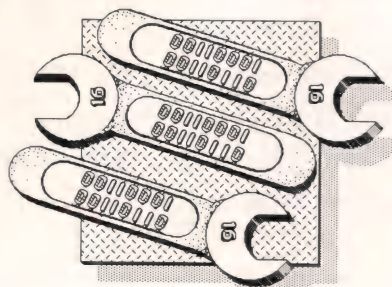
Finding this book among the flotsam and jetsam of computer publishing is like finding a \$100 bill in the street. The art, craft, and/or skill of effective debugging is a topic that has rarely been discussed in any useful way in the computer literature. The debugging chapters in manuals or textbooks are usually focused on defensive strategies and the design of test data, rather than on the stabilization, isolation, localization, and correction of bugs once detected. To new programmers, who have not yet developed a methodical approach to debugging, the process seems a little magical and their own approach is frequently haphazard and based largely on luck. Although native talent is an element of effective debugging, as it is of effective programming, a far larger component is simply the accumulated experience and the strategies learned by finding and fixing many different types of bugs in the past.

In his introduction, Mr. Ward writes: "Until I started teaching, I assumed that good debugging skills were a natural outgrowth of good design skills. Not so. A bright student may intuitively decompose a problem into beautifully coherent, cohesive, functional modules. That same student may not be able to find the

by Ray Duncan

most trivial syntax errors, let alone discover subtle runtime bugs. Equally bright students turn in working designs that literally defy analysis. While I don't believe that we learn debugging by studying design, I do believe that we can learn efficient debugging.

"We can develop a methodological



model that directs our efforts toward more productive searches. We can acquire heuristic knowledge (a kind of folk wisdom) about where to look first. We can be deliberately sensitive to the different variables and observable phenomena in different environments. We can become expert at selecting and using appropriate tools. And, through critical analysis of our attempts to find 'worthy' bugs, we can learn from our own mistakes."

The first chapters of *Debugging C* concentrate on the debugging process itself, with emphasis on recognition of bugs (lexical, syntactic, execution, and intent errors), their localization (using the principles of lexical, temporal, or referential proximity), a methodical approach, and good record keeping. Later chapters discuss the localization of compile-time errors and tracing methods. The last chapters of the book are particularly C-oriented, with excellent discussions of the special problems of C programmers: bugs due to data type mismatches, operator precedence or misuse, and uninitialized or out-of-range pointers. Here is where the author gently presents many useful tips that are common sense to veteran C coders, usually acquired through painful experience.

Example: "C, unlike Pascal and Basic, doesn't give an initial value to local variables when it creates them . . . Most stack frames are relatively small (less than 20 bytes) but every stack frame has a frame pointer (an address higher in the stack) and a return address (an address in the code area). Because uninitialized

pointers will use these obsolete addresses which occur frequently in reused areas of the stack, the programmer can expect 10 to 30 percent of the uninitialized pointers to reference code or areas of the stack." Similarly, Mr. Ward explains, with stack traces and memory dumps, why common errors such as initializing too many bytes of an automatic character array result in bizarre transfers of control or crashes "between lines" of source code.

The last few chapters describe the use of some common machine-level and symbolic debuggers, interpreters, and integrated development environments. The book's appendices include the C source code for a debugging subsystem that can be linked to a C application. The debugger provides tracing at several layers of granularity, stack frame displays, memory dumps, and watch points (periodic checks to see if a certain variable has been altered or has taken on a specified value).

The author couples a direct, lucid, and informal style of writing with a depth of understanding and highly structured presentation that are unusually effective. In spite of the title, most of the book has broad applicability and would be perfectly comprehensible to anyone with even a passing acquaintance with the C language.

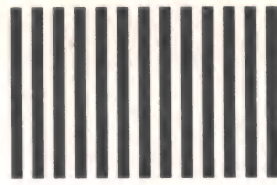
Hyman, Michael I. *Memory Resident Utilities, Interrupts, and Disk Management with MS and PC DOS*. Portland, Oreg.: Management Information Source, 1986. 373 pages with index. ISBN 0-943518-73-3.

This book, with the short catchy title, is anything but self-effacing. The introduction reads: "You will find this book to be the ultimate reference guide to getting the most out of your machine. As you read it, you'll learn how to use and enhance DOS to ex-



# For Free Info ...

NO POSTAGE  
NECESSARY  
IF MAILED  
IN THE  
UNITED STATES



## Start Here



Smart buyers start with *DDJ's* free information card, a shopping center filled with information about the products and services advertised in this very issue: everything from software and systems to peripherals and professional support services.

And smart buyers can use this free information card to quickly and easily gather a comprehensive file of facts, figures and product specs to sort out competing claims. Using *DDJ's* free information card can prevent you from making the wrong, costly buying decision.

Be a smart shopper. Complete and mail this postage paid card today!

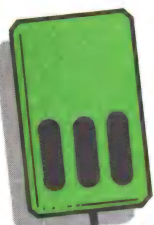


**BUSINESS REPLY MAIL**  
FIRST CLASS PERMIT #217, CLINTON, IOWA

POSTAGE WILL BE PAID BY ADDRESSEE

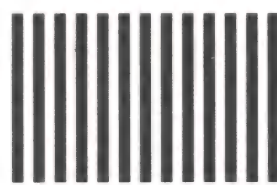
*Dr. Dobb's Journal of*  
**Software Tools**  
FOR THE PROFESSIONAL PROGRAMMER

P.O. Box 2157  
Clinton, Iowa 52735-2157



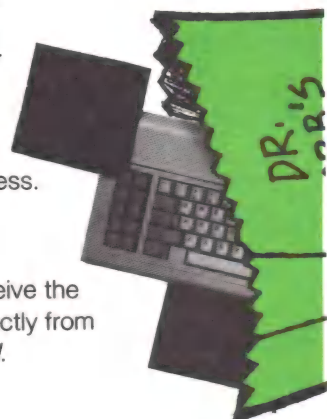
**Take a  
Reader  
Service  
Card  
with You**

NO POSTAGE  
NECESSARY  
IF MAILED  
IN THE  
UNITED STATES



## It's Easy as ...

- 1.** Circle the appropriate free information numbers, referring to the advertiser index for more information.
- 2.** Fill in your name and address.
- 3.** Mail today—postage is absolutely free. You'll receive the product information you need directly from the manufacturers, thanks to *DDJ*.



**BUSINESS REPLY MAIL**  
FIRST CLASS PERMIT #217, CLINTON, IOWA

POSTAGE WILL BE PAID BY ADDRESSEE

*Dr. Dobb's Journal of*  
**Software Tools**  
FOR THE PROFESSIONAL PROGRAMMER

P.O. Box 2157  
Clinton, Iowa 52735-2157



# Dr. Dobb's Journal of Software Tools

FOR THE PROFESSIONAL PROGRAMMER

Use this card for FREE, FAST information about the products and services listed in this issue. Simply circle the appropriate numbers below.

1	26	51	76	101	126	151	176	201	226	251	276	301	326	351	376
2	27	52	77	102	127	152	177	202	227	252	277	302	327	352	377
3	28	53	78	103	128	153	178	203	228	253	278	303	328	353	378
4	29	54	79	104	129	154	179	204	229	254	279	304	329	354	379
5	30	55	80	105	130	155	180	205	230	255	280	305	330	355	380
6	31	56	81	106	131	156	181	206	231	256	281	306	331	356	381
7	32	57	82	107	132	157	182	207	232	257	282	307	332	357	382
8	33	58	83	108	133	158	183	208	233	258	283	308	333	358	383
9	34	59	84	109	134	159	184	209	234	259	284	309	334	359	384
10	35	60	85	110	135	160	185	210	235	260	285	310	335	360	385
11	36	61	86	111	136	161	186	211	236	261	286	311	336	361	386
12	37	62	87	112	137	162	187	212	237	262	287	312	337	362	387
13	38	63	88	113	138	163	188	213	238	263	288	313	338	363	388
14	39	64	89	114	139	164	189	214	239	264	289	314	339	364	389
15	40	65	90	115	140	165	190	215	240	265	290	315	340	365	390
16	41	66	91	116	141	166	191	216	241	266	291	316	341	366	391
17	42	67	92	117	142	167	192	217	242	267	292	317	342	367	392
18	43	68	93	118	143	168	193	218	243	268	293	318	343	368	393
19	44	69	94	119	144	169	194	219	244	269	294	319	344	369	394
20	45	70	95	120	145	170	195	220	245	270	295	320	345	370	395
21	46	71	96	121	146	171	196	221	246	271	296	321	346	371	396
22	47	72	97	122	147	172	197	222	247	272	297	322	347	372	397
23	48	73	98	123	148	173	198	223	248	273	298	323	348	373	398
24	49	74	99	124	149	174	199	224	249	274	299	324	349	374	399
25	50	75	100	125	150	175	200	225	250	275	300	325	350	375	400

June '87: Use before September 30, 1987

Name \_\_\_\_\_  
 Title \_\_\_\_\_  
 Company \_\_\_\_\_ Phone \_\_\_\_\_  
 Address \_\_\_\_\_  
 City/State/Zip \_\_\_\_\_

# Dr. Dobb's Journal of Software Tools

FOR THE PROFESSIONAL PROGRAMMER

Use this card for FREE, FAST information about the products and services listed in this issue. Simply circle the appropriate numbers below.

1	26	51	76	101	126	151	176	201	226	251	276	301	326	351	376
2	27	52	77	102	127	152	177	202	227	252	277	302	327	352	377
3	28	53	78	103	128	153	178	203	228	253	278	303	328	353	378
4	29	54	79	104	129	154	179	204	229	254	279	304	329	354	379
5	30	55	80	105	130	155	180	205	230	255	280	305	330	355	380
6	31	56	81	106	131	156	181	206	231	256	281	306	331	356	381
7	32	57	82	107	132	157	182	207	232	257	282	307	332	357	382
8	33	58	83	108	133	158	183	208	233	258	283	308	333	358	383
9	34	59	84	109	134	159	184	209	234	259	284	309	334	359	384
10	35	60	85	110	135	160	185	210	235	260	285	310	335	360	385
11	36	61	86	111	136	161	186	211	236	261	286	311	336	361	386
12	37	62	87	112	137	162	187	212	237	262	287	312	337	362	387
13	38	63	88	113	138	163	188	213	238	263	288	313	338	363	388
14	39	64	89	114	139	164	189	214	239	264	289	314	339	364	389
15	40	65	90	115	140	165	190	215	240	265	290	315	340	365	390
16	41	66	91	116	141	166	191	216	241	266	291	316	341	366	391
17	42	67	92	117	142	167	192	217	242	267	292	317	342	367	392
18	43	68	93	118	143	168	193	218	243	268	293	318	343	368	393
19	44	69	94	119	144	169	194	219	244	269	294	319	344	369	394
20	45	70	95	120	145	170	195	220	245	270	295	320	345	370	395
21	46	71	96	121	146	171	196	221	246	271	296	321	346	371	396
22	47	72	97	122	147	172	197	222	247	272	297	322	347	372	397
23	48	73	98	123	148	173	198	223	248	273	298	323	348	373	398
24	49	74	99	124	149	174	199	224	249	274	299	324	349	374	399
25	50	75	100	125	150	175	200	225	250	275	300	325	350	375	400

June '87: Use before September 30, 1987

Name \_\_\_\_\_  
 Title \_\_\_\_\_  
 Company \_\_\_\_\_ Phone \_\_\_\_\_  
 Address \_\_\_\_\_  
 City/State/Zip \_\_\_\_\_

# For Free Info ...

## Start Here



Smart buyers start with DDJ's free information card, a shopping center filled with information about the products and services advertised in this very issue: everything from software and systems to peripherals and professional support services.

And smart buyers can use this free information card to quickly and easily gather a comprehensive file of facts, figures and product specs to sort out competing claims. Using DDJ's free information card can prevent you from making the wrong, costly buying decision.

Be a smart shopper. Complete and mail this postage paid card today!



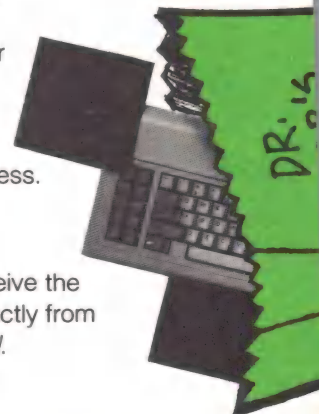
## Take a Reader Service Card with You

## It's Easy as ...

**1.** Circle the appropriate free information numbers, referring to the advertiser index for more information.

**2.** Fill in your name and address.

**3.** Mail today—postage is absolutely free. You'll receive the product information you need directly from the manufacturers, thanks to DDJ.





# The Advertiser Index

Advertiser Name	Page No.	Circle No.	Advertiser Name	Page No.	Circle No.
Aker Corporation	31	369	Oakland Group, Inc.	71	227
Aldebaran Laboratories	13	350	Oasys	35	254
Alpha Computer Service	81	321	Orchid Technology	15	130
Aptech Systems, Inc.	64	*	Oregon Software	131	357
Austin Code Works	107	250	Periscope Co. Inc.	110	214
BC Associates	87	182	Pharlap	91	343
Blaise Computing	2	159	Production Language Corp.	93	399
Blaise Computing	70	217	Programmer's Connection	67-69	129
Borland International	1	161	Programmers Shop (The)	40	133
Boston Software Works (The)	30	384	Programmers Shop (The)	41	301-304
Bryte Computer	78	387	Qualstar Corporation	97	356
Burton Systems Software	72	212	Quantum Computing	111	144
C Users Group	72	181	Quelo	84	377
C Toolbox	126	*	Quilt Computing	97	107
Compu View	125	122	Raima Corporation	103	*
CompuServe	65	237	68000 Toolbox	75	*
Cosmos, Inc.	26-27	400	SAS Institute	25	*
Creative Programming	91	*	Scantel Systems Limited Ltd.	99	391
Custom Software Systems	104	268	Scientific Endeavors	101	210
DDJ Subscriptions	36	*	Secom Information Products Co.	49	394
Datalight	9	203	Seidl Computer Engineering	88	114
Desktop A.I.	78	258	Semi-Disk Systems	133	85
Digital	130	127	SLR Systems	93	78
Ecosoft, Inc.	89	89	Softfocus	79	259
Essential Software	C3	138	Software Concepts Design	74	393
Fair-Com	44	93	Software Garden Inc.	18	314
Flexus	95	189	Software Security, Inc.	47	170
Genesis Data Systems	79	373	Softway	95	372
Gold Hill Computers, Inc.	20-21	*	Solution Systems	85	142
Greenleaf Software	48	97	Solution Systems	61	148
Guidelines Software	57	351	Solution Systems	61	149
Hauppauge Computer Works	C4	274	Solution Systems	38	152
Hi Tech Software	34	376	Texas Instruments	156	*
Integral Quality, Inc.	101	327	TRW - Command Support Division	59	*
Intel Corporation	53	190	TSF (The Software Family)	127	230
Kadak Products Ltd.	91	325	Tele Operating System	123	*
Kurtzberg Computer Systems	76	294	Turbo Tech	124	119
Lahey Computer Systems, Inc.	39	186	Unify Corporation	4	332
Language Processors, Inc.	77	266	Vermont Creative Software	32	157
Lattice, Inc.	129	101	W.R.I.S.T. Inc.	101	223
Lifeboat	45	118	Wendin	11	112
Logic Path	117	226	Workman & Associates	99	244
Logitech, Inc.	5	257	Xenosoft	84	225
Lugaru	24	135			
MSJ Subscriptions	100	*			
M&T Catalog of Books & Software Tools	*				
Magma Systems	66	313			
Magus Inc.	86	336			
Manx Software Systems	7	108			
Meridian Software Systems	33	397			
Metagraphics	80	392			
MetaWare Incorporated	63	95			
Micro Way	83	300			
MicroHelp, Inc.	19	215			
Microprocessors Unlimited	72	105			
Microsoft	118-119	380			
Microsoft	120-121	109			
Mortice Kern Systems, Inc.	90	249			
Nanosoft Associates	82	309			
Nantucket Corporation	135	220			
Norton Utilities (The)	24	243			

\*This advertiser prefers to be contacted directly;  
see ad for phone number.

## Advertising Sales Offices

### Southeast/Midwest

Gary George (404) 897-1923

### Northern California/Northwest

Lisa Boudreau (415) 366-3600

### Northeast

Cynthia Zuck (718) 499-9333

### Southern California/AZ/NM/TX

Michael Wiener (415) 366-3600

### Director of Marketing and Advertising

Ferris Ferdon (415) 366-3600



plore your computer and make mighty programs. Powerful examples will lead you along the way and provide models for later reference....” Such hype is commonplace from publishers’ PR departments, but it’s a little more unusual coming straight from the author’s pen. Several paragraphs farther on, you find the admonition “The letter ‘I’ and the number ‘1’ are represented in the program code by the same character. In no case is the letter ‘I’ used as a variable name. This should eliminate any confusion.” If there’s anything I’m already confused about at this point, it’s why the publisher of an “ultimate” reference book would allow it out of the door containing program listings wherein the numeral 1 and the letter I cannot be distinguished. But no matter, let’s see what the rest of the book has to offer.

Chapters 1 through 8 discuss the boot sector, file allocation table, directory structure, and file area of MS-DOS disks. Incredibly tortuous Turbo Pascal source code for a disk peeker/patcher named Explorer is developed as part of the exposition. Each chapter ends with a summary of sorts entitled “Key Programming Points.” Here are the Key Programming Points listed at the end of Chapter 4: “Sectors are the smallest organizational unit. They contain 512 bytes. You can find interesting messages and modify programs by editing sectors. Explorer uses arrow keys to move the cursor.”

The following few chapters discuss the partition table, “un-erasing” files, and a program that patches COMMAND.COM to change the names of MS-DOS internal commands (real useful!). Next the author covers input and output with the mouse, keyboard, and screen using IBM PC ROM BIOS drivers; an overview of file and record I/O; directory searching; and memory management. Nothing new here.

Finally, in Chapters 28 through 38, you get to the apparent reason for the book’s existence: the problems and pitfalls of programming Terminate and Stay Resident (TSR) utilities. This part of the book contains much useful information, poorly organized

and presented, about chaining onto interrupts, putting up and taking down pop-up displays, monitoring the keyboard for hot keys, and the like. If this part of the book were properly structured and edited, it would make a decent magazine article, but the book as a whole is a poor investment.

Norton, Peter; and Socha, John. *Peter Norton’s Assembly Language Book for the IBM PC*. New York: Brady/Prentice-Hall, 1986. 413 pages with index. ISBN 0-13-661901-0.

This book is Yet Another Assembly-Language Tutorial of average quality. It is built around the design and stepwise enhancement of a simple disk sector modification utility called DSKPATCH, discussing in passing some issues of processing keyboard input and updating screen displays using the IBM PC ROM BIOS video driver.

This book is mainly notable as a demonstration of the recent trend toward commodity marketing of computers and related products. As the competition in the wonderland of silicon has become more intense, we have seen the adoption of marketing tactics that were previously the province of car, appliance, and apparel manufacturers, such as scratch-off sweepstake tickets (Borland); cash rebates (Apple); “free gifts with purchase” (with every dBASE III Plus, get a Cross writing instrument free!); mystical mumbo jumbo such as Robert Carr, Wayne Ratliff, and Jonathan Sachs being touted as “Chief Scientists” of their respective companies; and last but not least, celebrity endorsements.

Back in 1983–1984, while John Socha was a contributing editor for *Softalk/PC*, he wrote a book called *Assembly Language Safari for the IBM PC: First Explorations* (Bowie, Md.: Brady, 1984. ISBN 0-89303-321-9). For various reasons, including rather poor production values in the book itself and massive financial problems at the Brady corporate level, the book received little attention and went out of print shortly thereafter. Nowadays, John Socha works for Peter Norton Inc. in Santa Monica, California, and is the author of the program sold as the Norton Commander. When *Peter Norton’s Assembly Lan-*

*guage Book* appeared, with Peter Norton and John Socha listed as coauthors, I thought it would be instructive to make a page-by-page comparison of John Socha’s old book with the new one.

A fairly generous assessment of the two books is that there are approximately 17 pages of new material in the “new” Norton/Socha version, scattered among the following topics: the *Proceed* command of DEBUG, a MASM program skeleton, MAKE files, SYMDEB, linker maps, .COM vs. .EXE programs, the *ASSUME* directive, segment overrides, and phase errors. The remainders of the two books are identical, except for some redrawn figures, the addition of some titles and divider pages, and some minor changes in wording that would have been introduced by any competent editor. There are also three new appendices that are mostly filler from other sources, such as MASM error messages and character tables. In other words, the substance of Mr. Norton’s contribution to this book seems to be his picture on the cover, his billing as a coauthor, and the running head “Peter Norton’s Assembly Language Book” on the top of each right-hand page.

Some might argue that the kind of misrepresentation involved here hurts no one and is therefore of no consequence. To be sure, the book’s purchasers, although they might have been misled about the creative origins of the book, have still bought a reasonably useful introduction to assembly language. The three principals—John Socha, Peter Norton, and Brady—are undoubtedly happy because their ploy has caused the book to vault onto the computer best-seller lists—which means everyone involved has made some money. And last but not least, Peter Norton’s recognition as an expert on all matters concerning the IBM PC has been magnified.

### A Poor Man’s MAKE

J. F. Philippe Marchand, of Webster, New York, sent in the programming goody of the month—a short program called CHKDATE.C. This program compares the date of two files and returns an “error level” code that can be tested within a batch file. For those people who are not fortu-



nate enough to have one of the commercial MAKE utilities (which are bundled with many of the compilers and assemblers being sold today), CHKDATE can help to automate the process of compiling and linking the various modules of an application program. Example 1, below, contains the C source code for the CHKDATE program, and Example 2, below, contains an example batch file that demonstrates the use of CHKDATE.

### MS-DOS Programming Tips

George Smith, of Lilburn, Georgia, writes: "I have run across one puzzling problem [in MS-DOS 3.0]. DOS function 0eh, which makes a specified drive the current drive, has al-

ways reported a reliable count of the number of drives on the system in register *al*. Under DOS 3.0 and later, though, the value returned in *al* is always at least 5, even if you happen to be running short-handed with fewer drives.

"I have no explanation, though I do have a solution in the form of function *DriveCnt* [Example 3, below]. The logic behind *DriveCnt* is simple. It takes the value *al* reports from function 0eh and passes it as a drive code to function 47h (Get Current Directory), which will set the carry flag if the drive code is invalid. It repeats calls to function 47h with successive lower drive codes until DOS

finds one it likes."

Although George has been kind enough to send in the routine *DriveCnt* for the edification of DDJ readers, he sells a package called Boosters for Turbo Pascal programmers that includes this subroutine and 77 others, a screen generator, some 40 example programs, and a 93-page manual. The Boosters package costs \$40 and can be ordered direct from George Smith at 609 Candlewick Lane, Lilburn, GA 30247; (404) 923-6879.

DDJ

Vote for your favorite feature/article.  
Circle Reader Service No. 9.

```
#include <stdio.h>
#include <sys types.h>
#include <sys stat.h>

main(argc,argv)
int argc;
char *argv[];
{ struct stat buf1, buf2;
  int result,k;
  if (argc < 3) {
    printf("chkdate: usage chkdate f1 f2 .. fn\n");
    printf("    will return errorlevel 1\n");
    printf("    if f1 older than f2 .. fn\n");
    printf("    or if f1 .. fn don't exist.\n");
    exit(1);
  };
  if (stat(argv[1], &buf1) != 0) exit(1);
  for (k=2; k<argc; k++) {
    if (stat(argv[k], &buf2) != 0) exit(1);
    if (buf1.st_atime < buf2.st_atime) exit(1);
  };
  exit(0);
}
```

**Example 1:** Phil Marchand's CHKDATE.C program

```
chkdate main.obj main.c
IF ERRORLEVEL 1 goto :compile1
goto :next1

:compile1
msc main,main;

:next1
chkdate func.obj func.c

IF ERRORLEVEL 1 goto :compile2
goto :next2

:compile2
msc func,func;

:next2
chkdate main.exe main.obj func.obj

IF ERRORLEVEL 1 goto :link
goto :exit

:link
link main.obj+func.obj,main.exe;

:exit
```

**Example 2:** TEST.BAT file, a demonstration of the use of the CHKDATE program in a batch file to automate the compilation and linking of an application

```
; FUNCTION DriveCnt : Integer;
; (C) 1986 George F. Smith & Company
;
; Purpose:
; Returns number of logical drives on host
; computer - DOS 2.0 and above.
;
; Sample Usage (Turbo Pascal):
; Writeln('Number of drives is ',DriveCnt);
;
; Suggested processing sequence:
; MASM DriveCnt,,,
; Link DriveCnt
; Exe2Bin DriveCnt DriveCnt.com
; C2I DriveCnt.com >DriveCnt.inl
;
; (C2I utility converts .com files to Turbo Pascal
; inline code. See DDJ, 10/86, pg. 90 )p
;
code segment
assume cs:code

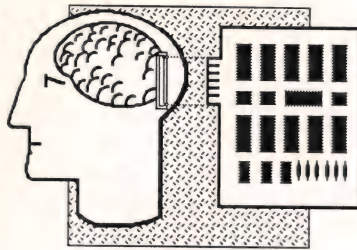
DriveCnt:
    push bp                ; standard
    mov bp,sp              ; subroutine
    push ds                 ; overhead
    sub sp,64               ; create scratch area,
    mov si,sp               ; save in si for 47h call
    mov ah,19h              ; get default drive
    int 21h                 ; returns drive code in al
    mov dl,al               ; move code to dl
    mov ah,0eh              ; set default drive to itself
    int 21h                 ; On return, al=# drives
;
; If DOS 3.0 or above, number of drives in al will
; be minimum of 5. Will repeat calls to function 47h
; until a valid drive code is obtained.
;
    mov dl,al               ; number of drives in dl
    push ss                 ; segment address
    pop ds                  ; of scratch buffer
search: mov ah,47h           ; is drive code okay?
    int 21h                 ; carry flag set if
    jnc okay                ; drive code invalid
    dec dl                  ; jump if code valid
    jnc okay                ; drive code too big;
    dec dl                  ; decrement it and
    jmp search              ; try again
okay:   xor dh,dh            ; clear dh
    mov [bp+4],dx           ; give results to caller
    add sp,64               ; adjust stack ptr
    pop ds                  ; restore caller's regs
    mov sp,bp
    pop bp
    ret
code ends

end DriveCnt
```

**Example 3:** George Smith's DriveCnt routine to determine the number of disk drives present in an MS-DOS



## Object-Oriented Programming in SCOOPS



**T**his month, I conclude my review of PC Scheme (which I consider to be the Turbo Pascal of the PC LISP family) with some examples of object-oriented programming using SCOOPS, the object-oriented extension of the language. To do this, I'll have to make some extensions to PC Scheme itself—but first, some discussion about LISP programming in general.

LISP is the most organic and lifelike of all programming languages. Most of its dynamic character comes from the combination of complex, nested structures with dynamic reassignment of structures of pointers and a simple syntax that uses the same representation for data and programming code.

The functional programming aspect of LISP involves a special implementation of argument passing so that usually few variables need to be stored permanently. The main thing in pure functional programming is not modifying objects in permanent storage but passing symbols as if they were values being passed between mathematical functions. The main result of such a program is the structure it returns rather than the state it creates in the permanent storage of the machine.

But LISP is not simply a single-paradigm programming language. It is a language that so far has been able to absorb each new programming model as it appears and to incorporate these models in a functioning whole. Over the years it has absorbed other programming concepts and is con-

da calculus.

In pure functional LISP programming, anything that does not just return a structure but modifies the machine is generally considered a side effect. But it is often important in an object-oriented environment to modify the object hierarchy dynamically in complex and carefully controlled ways. Take, for example, the case of performing simple list processing functions, such as updating and modifying list structures. Here it is often the case that either or both of the functions of returning the necessary structure and producing the necessary structure in permanent storage are important parts of the required tasks.

In the following example, I will show various versions of a function *add-to-end* that are implemented so as to return different values and produce different side effects. This function extends the list processing functions of LISP to include the ability to add an element to the end of an already existing list structure.

The function *give-n-take* was written to demonstrate the side effects of the *add-to-end* function. First, two lists are created—*nums*, which contains the list of number words (*one two three four five*); and *morenums*, which is composed of the complementary number word list (*six seven eight nine ten*). Here is what *give-n-take* looks like:

```
(define nums '(one two three four
                    five))
(define morenums '(six seven eight
                        nine ten))
(define (give-n-take)
  (add-to-end (car morenums) nums))
```

```
(set! morenums (cdr morenums)))
```

As you can see from the short session that follows, what *give-n-take* returns is different from the side effects it has on these lists. It simply returns the *morenums* list that was passed to it. When you ask LISP for the contents of these lists by typing their names at the interpreter prompt, however, you see the effects that *give-n-take* has each time it is called. It takes numbers successively from the beginning of the *morenums* list and adds them to the end of the *nums* list:

```
[2] nums
(ONE TWO THREE FOUR FIVE)
[3] morenums
(SIX SEVEN EIGHT NINE TEN)
[4] (give-n-take)
(SEVEN EIGHT NINE TEN)
[5] nums
(ONE TWO THREE FOUR FIVE SIX)
[6] morenums
(SEVEN EIGHT NINE TEN)
[7] (give-n-take)
(EIGHT NINE TEN)
[8] nums
(ONE TWO THREE FOUR FIVE SIX SEVEN)
[9] morenums
(EIGHT NINE TEN)
```

Now compare this to another version of the function called *add-to-end-2*. In the next session I create the list of integers from 1 to 4. My original *add-to-end* returns something unusable, but the side effects are the correct result. *Add-to-end-2* does just the opposite—it returns the list with the number added to the end, but when you examine the list of integers, you see that nothing has changed.

```
[6] (define integers '(1 2 3 4))
INTEGERS
[7] (add-to-end 5 integers)
(4 5)
[8] integers
```

by Ernest R. Tello

tinuing to do so—for example, as I mentioned last month, SCOOPS has assimilated some features of Smalltalk. The original model on which the LISP language was based was that of functional programming, using the lamb-



(1 2 3 4 5)  
[9] (add-to-end-2 6 integers)  
(1 2 3 4 5 6)  
[10] integers  
(1 2 3 4 5)

This hasn't been just an academic exercise. The side-effects version of *add-to-end* is useful for doing necessary housekeeping in an object-oriented LISP environment. It is valuable to be able to maintain lists of all the current instances of various classes that are alive in a system that is changing dynamically. Without the version of *add-to-end* that can actually modify such lists, you would not be able to update them continually.

Another function that could be useful is *delete-last!*. It performs the opposite service—that of destructively removing the final element in a list. Its definition is:

```
(define (delete-last! lst)
  (delete! (car (last-pair lst)) lst))
```

The *last-pair* function in PC Scheme returns the last pair in a list. The function works for returning the last element as a single element list because the *car* reduces the pair to a simple list structure.

The following quick session shows the behavior of *delete-last!*. As you can see, in this case both what the function returns and its side effects are identical.

```
[2] (define numbers '(one two three
                        four five))
NUMBERS
[3] (delete-last! numbers)
(ONE TWO THREE FOUR)
[4] numbers
(ONE TWO THREE FOUR)
```

### Programming in SCOOPS

Sending messages is a rather simple matter of using the *send* function with the *receiver* object and the message to be sent plus its arguments. So, to send a message to the *my-body* object (introduced in the May column), giving it a new body part called *toes*, you would say:

```
[1] (send my-body put-cpart-name
        'toes)
[2] body-parts
(HEAD NECK ARMS HANDS TRUNK LEGS
 FEET TOES)
```

If you then changed your mind and decided to remove this new body part, you could do so by globally accessing the *body-parts* list using the newly defined function *delete-last!* as follows:

```
[3] (delete-last! body-parts)
[4] body-parts
(HEAD NECK ARMS HANDS TRUNK LEGS
 FEET )
```

The code in Listing One, page 98, demonstrates simple inheritance in PC Scheme through several levels of a fairly linear hierarchy. First, the root class *artifact* is defined with the instance variables *material*, *weight*, *purpose*, and *cost*. Then *transport-means* is defined as a subclass of *artifact* with the additional instance variables *medium*, *time-range*, and *power-source*. Naturally, *transport-means* inherits all the variables and methods of the *artifact* class. Then *transport-vehicle* is defined as the next subclass and *passenger-vehicle* as a subclass of it. Descending further in the same linear manner of adding more and more specific classes that inherit everything from the previous class, the classes *water-transport-vehicle*, *surface-vessel*, *ship*, and *ocean-liner* are defined. The instance *object ship1* is created as an instance of the class *ship*. The two methods *speed* and *direction* are also provided for the *ship* class.

Listing Two, page 99, provides an example of multiple inheritance in SCOOPS. Here I have implemented the example used in April's column in PC Scheme. First, the classes *business* and *adversary* are defined. Then the class *competitor* is defined and uses the multiple inheritance feature to inherit everything from both of these two classes.

### Availability

All the source code for articles in this issue is available on a single disk. To order, send \$14.95 to *Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94063 or call (415) 366-3600 ext. 216. Please specify the issue number and format (MS-DOS, Macintosh, Kaypro).

DDJ

(Listings begin on page 98.)

Vote for your favorite feature/article.  
Circle Reader Service No. 10.

# DIRECTLY ACCESS dBASE III DATA FILES

## db DOS \$39.95

CREATE, VIEW AND EDIT dBase III data files from the DOS prompt. Use fast and powerful full screen editing with search, append and direct GOTO capabilities. Output to the screen or printer at your choice. All this from the DOS prompt.

## db PASCAL \$29.95

TURBO CHARGE YOUR dBASE III FILE ACCESS. Use dBase III data files with Turbo Pascal. Simple dBase III file access. Report output with turbo speed.

## ORDER NOW

Phone orders

**1-800-433-6854**

In Arizona

**(602) 435-2370**

Or send check or money order to:

LogicPath

P.O. Box 1267

Chandler, AZ 85224-1267.

We welcome VISA/MC or COD in certified US funds only. Add \$2.50 for shipping per order. In Arizona add 6% sales tax. Call or write for other products or additional information (602) 435-2370.

# LOGICPATH

P.O. Box 1267 • Chandler, AZ 85244-1267  
Turbo Pascal, dBase III trademarks of Ashton-Tate.

CIRCLE 226 ON READER SERVICE CARD



# The fastest C

Your search for execution speed is over. The new Microsoft® C Compiler Version 4.0 is here. With blazing performance. We've added common sub-expression elimination to our optimizer that produces code that rips through the benchmarks faster than ever before.

---

"...the Microsoft performance in the benchmarks for program execution is the best of the lot overall."  
— William Hunt, *PC Tech Journal*, January, 1986.\*

---

But speed isn't the only edge you get with Microsoft C. Other advantages include a variety of memory models like our new HUGE model that breaks the 64K limit on single data items. Plus our NEAR, FAR and HUGE pointers, which provide you greater flexibility. All this allows you to fine tune your program to be as small and fast as possible.

---

"Excellent execution times, the fastest register sieve, and the best documentation in this review ... Microsoft Corporation has produced a tremendously useful compiler." — Christopher Skelly, *Computer Language*, February, 1986.

---

## No more debugging hassles. Introducing CodeView. Free.

Now, for a limited time, we'll give you an unprecedented programming tool when you buy Microsoft C, free. New Microsoft CodeView™ offers the most powerful tool yet in



the war on C bugs. Forget the hex dumps. Now you can view and work with programs at any level you want. Use the program source, the disassembled object code, or

### Microsoft C Compiler Version 4.00

#### Microsoft C Compiler

- Produces fast executables and optimized code including elimination of common sub-expressions. **NEW!**
- Implements register variables.
- Small, Medium and Large Memory model libraries.
- Compact and HUGE memory model libraries. **NEW!**
- Can mix models with NEAR, FAR and the new HUGE pointers.
- Transport source and object code between MS-DOS® and XENIX® operating systems.
- Library routines implement most of UNIX™ System V C library.
- Start-up source code to help create ROMable code. **NEW!**
- Full proposed ANSI C library support (except clock). **NEW!**
- Large number of third party support libraries available.
- Choose from three math libraries and generate in-line 8087/80287 instructions or floating point calls:
  - floating point emulator (utilizes 8087/80287 if installed).
  - 8087/80287 coprocessor support.
  - alternate math package — extra speed without an 8087/80287.
- Link your C routines with Microsoft FORTRAN (version 3.3 or higher), Microsoft Pascal (version 3.3 or higher) or Microsoft Macro Assembler.
- Microsoft Windows support and MS-DOS 3.1 networking support.
- Supports MS-DOS pathnames and input/output redirection.

#### Microsoft Program Maintenance Utility. **NEW!**

- Rebuilds your applications after your source files have changed.
- Supports macro definitions and inference rules.

#### Other Utilities

- Library Manager.
- Object Code Linker.
- EXE File Compression Utility.
- EXE File Header Utility.

#### C Benchmarks

In seconds

	Microsoft C 4.0	Lattice C 3.0	Computer Innovation C 2.3	Aztec C86 3.2	Wizard C 3.0
Sieve of Eratosthenes (register)	82.9	151.4	172.3	88.0	91.9
Copy Block	86.9	231.7	199.0	123.8	189.5

Run on an IBM PC XT with 512K memory

#### Microsoft CodeView

#### Window-oriented source-level debugger. **NEW!**

- Watch the values of your local and global variables and expressions as you debug.
- Set conditional breakpoints on variables, expressions or memory; trace and single step.
- Watch CPU registers and flags as you execute.
- Effectively uses up to four windows.
- Debug using your original source code, the resulting disassembly or both intermingled.
- Use drop-down menus to execute CodeView commands.
- Access the on-line help to lead you through CodeView's options and settings.
- Easily debug graphics-oriented programs since program output is kept separate from debugger output.
- Keyboard or optional mouse support.
- Enter in familiar SYMDEB or DEBUG commands.



# you've ever seen.

both at the same time. Open a window to view CPU registers and flags. Watch local and global variables as well. All while your program is running.

CodeView gives you complete control. Trace execution a line at a time — using source or assembly code. Or set conditional breakpoints on variables, memory or expressions. CodeView supports the familiar SYMDEB command syntax, as you'd expect. Commands are also available through drop-down menus. Combine the new window-oriented interface with our on-line help and debugging has never been easier. Or quicker.

## Take the \$5 CodeView tour.

You may find it hard to believe our debugger can do all we've claimed. So we're offering test drives. Five bucks will put you behind the wheel of a Microsoft C demo disk with CodeView.\* See for yourself how fast debugging can get.

For more information about the CodeView demo disk, the new Microsoft C Compiler, a list of third party library support or the name of your nearest Microsoft dealer, call (800) 426-9400. In Washington State and Alaska, (206) 882-8088. In Canada call (416) 673-7638.

```

File  Search  View  Run  Watch  Options  Calls  Trace!  Go!
-----
                                math.c
0) island : 244
1) tiszero() : 1
2) 4034:0000 00 00 00 00 00 00 00 00 43 72 .....

3DB5:00EE B80200      MOV     AX,0002
3DB5:00F1 E89402      CALL    _chkstk (0388)
3DB5:00F4 56          PUSH    SI
3DB5:00F5 8B7604      MOV     SI,Word Ptr [BP+04]
13:          t[0] = 1;
3DB5:00F8 C06441A01      MOV     Byte Ptr [t (1A44)],01
14:          div(s);          /* t[] = 1/s */
3DB5:00FD 56          PUSH    s
3DB5:00FE E82601      CALL    _div (0227)
3DB5:0101 83C402      ADD     SP,+02
15:          add();
3DB5:0104 E84D00      CALL    _add (0154)          :BR0
16:          island = 1;
3DB5:0107 C746FE0100      MOV     Word Ptr [island],0001
17:          do {

>da 33 0x29
4034:0021 Microsoft
>

```

AX = 0002  
 BX = 310C  
 CX = 00F4  
 DX = 1E80  
 SP = 31CA  
 BP = 31CE  
 SI = 0002  
 DI = 32A8  
 DS = 4034  
 ES = 4034  
 SS = 4034  
 CS = 3DB5  
 IP = 00F8  
 overflow  
 up  
 enable  
 positive  
 not zero  
 no auxcy  
 odd  
 carry

## Microsoft® C Compiler

The High Performance Software

Microsoft, MS-DOS and XENIX are registered trademarks and CodeView is a trademark of Microsoft Corporation. UNIX is a trademark of AT&T Bell Laboratories. IBM is a registered trademark of International Business Machines Corporation. \*Offer expires 12/31/86.



# The fastest, tightest code.



## (Though the same can hardly be said of the name.)

We have to tell you, we had a hard time getting the name down this short.

Because Microsoft's new FORTRAN Compiler actually has a far longer list of features.

It uses the same optimizer and code generator technology that made our C Compiler the industry leader.

And we've also added special loop optimizations that give you the

smallest, fastest FORTRAN code a PC can handle.

*"Now Microsoft's FORTRAN Optimizing Compiler generates such fast code that an IBM PC/XT approaches the speed of the VAX."*

*Peter Osgood, MIT, Project Athena, Director of the Real Time Lab Project.*

This compiler has already passed the toughest test there is. It's been

### Microsoft FORTRAN Optimizing Compiler Version 4.0.

- ◆ Uses the Microsoft C optimizing technology, plus loop optimization to generate the fastest executable code for MS-DOS. NEW!
- | Execution Speed<br>(in Seconds) | Microsoft<br>FORTRAN<br>v. 4.0 | Ryan-McFarland<br>FORTRAN<br>v. 2.11 | IBM Professional<br>FORTRAN<br>v. 1.22 |
|---------------------------------|--------------------------------|--------------------------------------|--|
| Sieve                           | 7.97                           | 9.33                                 | 38.51                                  |
| Whetstone                       | 53.82                          | 58.67                                | 79.04                                  |
| Lookup                          | 5.82                           | 18.61                                | 26.02                                  |
- ◆ Fully GSA certified for ANSI 77 compatibility with no errors at the highest level. NEW!
  - ◆ Numerous IBM VS and DEC VAX extensions. NEW!

- ◆ Microsoft CodeView: Window-oriented source-level debugger. NEW!
  - Debug using your original source code, the resulting disassembly or both intermingled.
  - Watch and change the values of your local and COMMON variables as you debug.
  - Set conditional breakpoints on variables, expressions or memory; trace and single step.
  - Debug Microsoft C programs as well as Microsoft Fortran programs.
  - Watch and change registers and flags as you execute.
  - Easily debug graphics oriented programs since program output is kept separate from debugger output.



GSA-certified as Full ANSI FORTRAN 77, and 100% error-free.

*"The Microsoft FORTRAN Optimizing Compiler let us port the 200,000 line Boeing Mathematical Library (BCSLIB) with virtually no changes. This ANSI FORTRAN 77 code was ported directly from Cray, CDC, DEC, IBM and other mainframes and workstations."*

*Ivor Philips, Boeing Computer Services, Program Manager  
Mathematical Software Libraries.*

We've also included the same advanced intrinsic math functions found on VAX® and IBM® VS systems. Add

Compiler v. 4.0 with CodeView™

improvements like our new HUGE memory model, and porting the biggest mainframe programs has never been easier.

Among the many additions we've made to our package is our exclusive CodeView™ windowing debugger.

It lets you trace through programs at any level you want, from source code to assembly language.

You can open windows and watch both variables (local and COMMON) and CPU registers change.

You can set conditional breakpoints using variables and expressions.

Debugging gets even easier with the compiler's advanced diagnostics. Detailed error messages are thoroughly explained and cross-referenced in our new manuals.

Documentation that has been completely revised and expanded with tons of examples.

If we're talking your language, use one of the numbers below for more details about the Microsoft® ANSI FORTRAN 77 Optimizing Compiler

Version 4.0 with CodeView, and the name of your nearest dealer.

(Even if the call's toll-free, it may be a good idea to refer to it as "FORTRAN 4.0" for short.)

## Microsoft® FORTRAN

The High Performance Software.

Call (800) 426-9400. In Washington State or Alaska, (206) 882-8088. In Canada, (416) 673-7638.

Microsoft and MS-DOS are registered trademarks and CodeView is a trademark of Microsoft Corporation. IBM is a registered trademark of International Business Machines Corporation. VAX is a registered trademark of Digital Equipment Corporation.

- ◆ Medium, Large and Huge Memory Model Libraries. NEW!
- ◆ Mix models with NEAR, FAR and new HUGE pointers.
- ◆ Common blocks and arrays greater than 64K.
- ◆ Choose from three math libraries and generate in-line 8087/80287 instructions or floating point calls:
  - floating point emulator (utilizes 8087/80287 if installed)
  - 8087/80287 coprocessor support
  - alternate math package—extra speed without an 8087/80287
- ◆ Link your FORTRAN routines with Microsoft C (v. 4.0 or higher), Microsoft Pascal (v. 3.3 or higher) or Microsoft Macro Assembler.
- ◆ Largest number of 3rd party support libraries available.

- ◆ Provides more detailed diagnostic error messages (almost twice as many as competitors) and extensive documentation with non-ANSI 77 features highlighted. NEW!
- ◆ Proven reliability—tested with over 2.5 million lines of code compiled and executed.
- ◆ MS-DOS® network support with file / record locking and sharing.
- ◆ Microsoft Program Maintenance Utility rebuilds your applications after your source files have changed. NEW!
- ◆ Other utilities including faster overlay linker (links over 1Mbyte object code), library manager, EXE file compression utility, EXE file header utility, MS-DOS environment setting utility and setup utility.



fastest and just about the handiest editor under ten fingers (if it isn't, just redefine your macros the instant you think of something better). Just take the compiler and all-purpose resident macro program as bonuses.

Now, the importance of speed is simply that we want to work at programming, not at running an editor. Like a fine sound system, a good editor should be transparent. When I write *while v <= maxv do*, I don't think of the keys I press, only of the statement I'm producing. It should be the same when I want to move that statement, to indent it and what follows, or simply to erase the line—or to find it in a 2,000-line file. What no one wants is to sit and look at the editor editing.

Which brings us to WordStar imprinting. I got imprinted with WordStar because my mother imprinted

me with ten fingers. For the first two days, it may be easier to use an editor that has F3 for "delete word" and shift-F3 for "delete line," but after those two days, only a WordStar-style editor gives you a chance to get to the point at which you only have to think "delete line" and not notice the actual fingerwork needed to do it—because it doesn't demand that your hands leave the typing position.

Hunt-and-peck chickens may not understand this, but there is a world beyond the barnyard, you know. The WordStar commands aren't meant for the user's manual but for the user's hands. They're the natural extension of sheer speed because they remove another nontransparent interface between your mind and the program text.

So, my ultimate editor is simply the Turbo Pascal editor with a good mac-

ro program plus several-document capacity, windowing, wordwrap for comments, block-limited search-and-replace with more complex specification capacity, and files greater than 64K. But never at the price of speed or the WordStar keyboard code. Don't mistake us far-voyaging mallards for clay pigeons.

Philippe Ranger

6120 Hutchison

Montreal, Canada H2V 4C2

## 6502 Hacks

Dear DDJ,

When I read Mark Ackerman's "6502 Hacks" (February 1987), I was both surprised and delighted that, in this world of 68000s and hypercubes, there is still anyone left who would spend the time and effort to write a good article about 6502 programming.

But, before anyone's programs start crashing, let me correct Mr. Ackerman on two related points. First, the software interrupt instruction (*BRK*) uses the *IRQ* (maskable interrupt) vector, not the *NMI* (nonmaskable interrupt) vector. But to make matters even worse, the *BRK* instruction pushes its address+2 on the return stack. In order to return to the opcode immediately following the *BRK*, the return address on the stack must be dug up and decremented before returning (very messy).

Second, and more important, is that the *RTI* (return from interrupt) instruction is not functionally compatible with the sequence *PLP* (pull processor), *RTS* (return from subroutine). The *RTI* instruction pulls the processor status byte and then continues execution at the address pulled from the stack, increments the address by 1, and then continues execution at the adjusted address. This compensates for the fact that the *JSR* pushes the address of the next opcode-1.

The reason for this lies in a quirk of the 6502 opcode processing. To execute the *JSR* instruction, the processor first fetches the opcode and the low byte of the address. It then pushes the current program counter (which is now pointing to the third byte of the instruction) and, finally,

```
ENTRY      STX   SAVEX      ;Save [X]
           ...           ;(Body of subroutine)
           LDX   #$FF       ;Restore [X]
SAVEX      EQU   *-1        ;Data field of preceding instruction
           RTS
```

### Example 1: A fast method for saving and restoring registers

```
TXBLOCK    STX   _TXLENGTH   ;Set block length (0=256)
           STA   _TXADDR     ;Set block address
           STY   _TXADDR+1
           LDX   #-1         ;Init timeout
           LDY   #0          ;Init block index
           STY   _TXSUM      ;Reset TXSUM accum.
TXBLOOP1    LDA   $FFFF,Y     ;First, get byte to send
           EQU   *-2         ;Address portion of LDA
                           ;instruction
TXBLOOP2    BIT   HWREADY     ;Check if Tx port is ready
           BPL   TXBLOOP3    ;(N)Check timeout
           STA   HWDATA       ;(Y)Tx the byte
           EOR   #$FF         ;Accum TXSUM
           EQU   *-1         ;Data portion of EOR instruction
           STA   _TXSUM      ;Update running XOR sum
           INY
           CPY   #$FF        ;End block?
           EQU   *-1         ;Data portion of CPY instruction
           BNE   TXBLOOP1    ;(N)Continue sending
           LDA   _TXSUM       ;Send the checksum
           ...
TXBLOOP3    DEX              ;Timeout expired?
           BNE   TXBLOOP2    ;(N)Continue sending
           ...              ;(Y)Return timeout error
```

### Example 2: An actual 6502 code fragment



fetches the high byte of the address. So, the address that gets pushed on the stack is always one byte shy of the next instruction. In contrast, the interrupt acknowledge sequence will only happen between instructions, so the program counter is always pointing to an opcode when the interrupt return address is pushed.

In his coverage of self-modifying code, I think Mr. Ackerman missed one very useful trick. I often save registers (because they are so very scarce) by storing their contents in the data portion of a load instruction located at the end of the routine [see Example 1, page 122]. It takes 5 bytes of code but is absolutely the fastest method of saving and restoring a register (six cycles total) and both the instruction and data storage are localized in the subroutine.

To demonstrate variations of the trick, I have included a programming fragment of an actual routine [see Example 2, page 122]. The routine transmits, over an extremely fast synchronous communication port, a block of data (from 1 to 256 bytes) followed by the exclusive-ORed sum of the block. It must also make sure that the port is not "hung" by keeping a decaying timer in the X register. The routine is passed the address of the block in registers A and Y and the length of the block in X. When the loop is cooking, it can transmit a byte every 27 cycles.

Of course, I won't tell you what product this code is running in (the labels have been changed to protect the innocent), for fear that someone wouldn't buy our system if they knew I programmed this way on a regular basis.

James Bucanek  
C-Si Systems  
572 W. Pima  
Coolidge, AZ 85228

DDJ

# The Tele Operating System Toolkit

The unique features of this four part, multitasking operating system will allow you to fully exploit the power of any 8086-based machine! **Tele** is written in C and assembly language for IBM PC compatibles and includes **preemptive multitasking capabilities and an unlimited number of tasks**. Tele contains full C and **Assembler source code**, as well as precompiled libraries. It is compatible with MS-DOS, Unix, and the MOSI standard. MS-DOS disk format.

**SK: The System Kernel** includes the most crucial part of the Tele Operating System - the preemptive multitasking algorithm. **SK** also contains an initialization module, general purpose utility functions for string and character handling, format conversion, terminal support and machine interface, along with a real-time task management system. All other components require SK: The System Kernel.

SK

Item #090

\$49.95

**DS: The Display Driver** contains BIOS level drivers for a memory-mapped display (the fastest way to display data), window management support and communication coordination between the operator and tasks in a multitasking environment. **DS** includes functions to create and delete virtual displays, and functions to overlay a portion of a virtual display on the physical display.

An unlimited number of virtual displays can belong to any particular task, and an unlimited number can be in the system at any time. Requires SK: The System Kernel.

DS

Item #091

\$39.95

Dr. Dobb's Journal of  
**Software Tools**  
FOR THE PROFESSIONAL PROGRAMMER



**SK:**  
The System Kernel  
of the Tele Operating  
System Toolkit

by Ken Berry

Dr. Dobb's Journal of  
**Software Tools**  
FOR THE PROFESSIONAL PROGRAMMER



**DS:**  
Window  
Display

by Ken Berry

**To Order:** Return this order form with your payment to M&T Books, 501 Galveston Dr., Redwood City, CA 94063 Or, Call **TOLL-FREE 800-533-4372** Mon-Fri 8AM-5PM. In CA call **800-356-2002**.

**YES!**

Please  
send me:

Item #090 SK: The System Kernel \$49.95

Item #091 DS: The Display Driver \$39.95

Subtotal

CA Residents add tax

Add \$2.25 per item for shipping

TOTAL

☐ Check Enclosed. **Make Payable to M&T Publishing, Inc.**

Charge my

☐ VISA

☐ M/C

☐ Amer. Exp

Card #

Exp.

Name

Address

City

State

Zip

31283



## VIEWPOINT

(continued from page 14)

ple) to use simple trigonometry in a high-level language than it is to use it in assembly language.

High-level languages can enhance the value of a programmer's work by allowing code to run with very little alteration on many different computers—all that is required is a compatible compiler for each. My 68000 cross assembler is now operational on computers that use Z80, 8088, and 68000 processors. I would like to challenge Suman—or any assembly-language programmer—to convert an assembly-language program of similar functionality and complexity to run in two environments. I suspect that it would take them longer than the few days each that it took to port the 68000 cross assembler I presented in the April and May 1986 issues of *DDJ*.

With reference to Suman's specific criticisms, the use of named constants (for any language, including assembly language) is almost universally considered to improve maintainability by localizing changes. Al-

though I used the constants *FIRST* and *LAST* only twice (not once, as Suman stated), I feel their use was justified.

As to the lack of initialized variables in Modula-2, I agree that this is a defect of the language. That should not, however, be an indictment of all high-level languages—C does allow initialized variables, including initialized structures and arrays. If Modula-2 allowed initialized variables, there would have been no need for *InitOperationCodes* (the module that Suman disliked so much). The mnemonic lookup table could have been created at compile time—in the module that needed the table—and with much less overhead.

Certainly, the mnemonic lookup table of X68000 could have been written more efficiently using assembly language. It would have been harder to write or to read, however, and it would not have been portable (some processors invert the order of bytes within words). The sets that were used for the allowable addressing modes (*ModeA* and *ModeB*) repre-

sented the 68000 addressing scheme in a recognizable way. For example, one of the set members was called *Size67* to indicate that the sixth and seventh bits of the 68000 operation code were used to indicate the size of the operation. If that member were to be converted to a computer word (1s and 0s), a programmer reading the listing might be at a loss to figure out what the significance of a particular bit (or combination of bits) was.

Finally, Suman seems to contradict his own point about turgidity and redundancy when he suggests that I should have used 118 individual *write* statements instead of the simple loop:

```
FOR i := FIRST TO LAST DO
  WriteRec (f, Table68K[i]);
END;
```

Perhaps an ideal solution to the dilemma of choosing between the expressiveness of high-level languages and the efficiency of assembly language is to mix the two. Most decent compilers allow the integration of assembly-language modules. A program can be written and debugged in Modula-2 or C, then profiled to identify the bottlenecks. Finally, these sections can be rewritten in assembly language. The original high-level language code can be left behind as a comment block to the assembly language. Although this would hamper portability somewhat, the sections of a program that benefit from recasting in assembly language are usually a small portion of the overall code. Projects that would be Herculean tasks in assembly language become quite comfortable using modern high-level languages. Get your project working, then if you need more speed or better memory utilization, tune it up using a small amount of hand-coded assembly language.

DDJ

Vote for your favorite feature/article.  
Circle Reader Service No. 1.

## IS THERE A VOID IN YOUR LIFE?

Admit it — you've been missing something. What you need is a practical publication that speaks Turbo Pascal, and Turbo Pascal only.

*Turbo Tech Report* does just that: this bimonthly newsletter disk publication provides practical programming solutions both on disk and on paper. The approach is hands-on and how-to. Every issue contains:

- Articles written by Turbo Pascal experts.
- Reviews of the latest, hottest Turbo Pascal software products.
- Practical demonstrations of how to solve a particular problem with a Turbo Pascal product.
- A disk filled with code. You'll receive applications developed by authors, plus useful utilities, libraries and routines from Turbo Pascal users worldwide.

Fill the void with a subscription to *Turbo Tech Report*: 6 issues with 6 disks for \$99. Call (800) 533-4372 or Calif. residents call (800) 356-2002 and start your subscription today!





## #1 PROGRAMMABLE EDITOR

(Call for FREE DEMO disk)

```
TEXT LINE: 15 COL: 16 FILE: PHOTO .203 INSERT E1
WINDOW 0
/*= Main loop - displays the ma
do {
  scrlines = SCRINES:
  scrwidth = SCRWIDTH:
  clrscrn(scrlines-20);
  show( main_menu );
  ret_val = getrange( mn_pro
  process( ret_val, (new_ved
} while ( ret_val != EXIT_OK )

if (new_vedit && (table_in !=
  printf( crt_sel );
  if (yesno( " " )) setcrt( ar
  else outcrlf();
}
=WINDOW $

DIRECTORY C:\VEDIT\NEW
COMPARE .VDM CU203 .VDM MAIL .VDM MENU .VDM PRINT .VDM
SORT .VDM STRIPV .VDM Z80-8086.VDM
```

VEDIT PLUS is an advanced editor that makes your program development and word processing as efficient and easy as possible. VEDIT PLUS is simple enough to learn and use for the novice, yet has the speed, flexibility and power to satisfy the most demanding computer professional. VEDIT PLUS is particularly suited for writing all types of programs and lengthy documents such as reports or manuscripts.

.sp 2  
This shows how VEDIT PLUS can perform windowing. One window is used for word processing, a second for program development, and the third for commands. Up to 40 windows are supported and you determine each window's size and color.

Stunning speed. Unmatched performance. Total flexibility. Simple and intuitive operation. The newest VEDIT PLUS defies comparison.

### Try A Dazzling Demo Yourself.

The free demo disk is fully functional - you can try all features yourself. Best, the demo includes a dazzling menu-driven tutorial - you experiment in one window while another gives instructions.

The powerful macro programming language helps you eliminate repetitive editing tasks. The impressive demo/tutorial is written entirely as a 'macro' - it shows that no other editor's 'macro' language even comes close.

Go ahead. Call for your free demo today. You'll see why VEDIT PLUS has been the #1 choice of programmers, writers and engineers since 1980.

Available for IBM PC, TI Professional, Tandy 2000, DEC Rainbow, Wang PC, MS-DOS, CP/M-86 and CP/M-80. (Yes! We support windows on most CRT terminals, including CRT's connected to an IBM PC.) Order direct or from your dealer. \$185.

### Compare features and speed

	BRIEF	Norton Editor	PMATE
Off the cuff macros	No	No	Yes
Built-in macros	Yes	No	Yes
Keystroke macros	Only 1	No	No
Multiple file editing	20 +	2	No
Windows	20 +	2	No
Macro execution window	No	No	No
Trace & Breakpoint macros	No	No	Yes
Execute DOS commands	Yes	Yes	Yes
Configurable keyboard			
Layout	Hard	No	Hard
Cut and paste buffers	1	1	1
Undo line changes	Yes	No	No
Paragraph justification	No	No	No
On-line calculator	No	No	No
Manual size / index	250/No	42/No	469/Yes
Benchmarks in 120K File:			
2000 replacements	1:15 min.	34 sec.	1:07 min.
Pattern matching search	20 sec.	Cannot	Cannot
Pattern matching replace	2:40 min.	Cannot	Cannot

### VEDIT PLUS FEATURES

- Simultaneously edit up to 37 files of unlimited size.
- Split the screen into variable sized windows.
- 'Virtual' disk buffering simplifies editing of large files.
- Memory management supports up to 640K.
- Execute DOS commands or other programs.
- MS-DOS pathname support.
- Horizontal scrolling - edit long lines.
- Flexible 'cut and paste' with 36 'scratch-pad' buffers.
- Customization - determine your own keyboard layout, create your own editing functions, support any screen size.
- Optimized for IBM PC/XT/AT. Color windows. 43 line EGA.

### EASY TO USE

- Interactive on-line help is user changeable and expandable.
- On-line integer calculator (also algebraic expressions).
- Single key search and global or selective replace.
- Pop-up menus for easy access to many editing functions.
- Keystroke macros speed editing, 'hot keys' for menu functions.

### FOR PROGRAMMERS

- Automatic Indent/Undent for 'C', PL/I, PASCAL, etc.
- Match/check nested parentheses, i.e. '{' and '}' for 'C'.
- Automatic conversion to upper case for assembly language labels, opcodes, operands with comments unchanged.
- Optional 8080 to 8086 source code translator.

### FOR WRITERS

- Word Wrap and paragraph formatting at adjustable margins.
- Right margin justification.
- Support foreign, graphic and special characters.
- Convert to/from WordStar and mainframe files.
- Print any portion of file; selectable printer margins.

### MACRO PROGRAMMING LANGUAGE

- If-then-else, looping, testing, branching, user prompts, keyboard input, 17 bit algebraic expressions, variables.
- Flexible windowing - forms entry, select size, color, etc.
- Simplifies complex text processing, formatting, conversions and translations.
- Complete TECO capability.
- Free macros: • Full screen file compare/merge • Sort mailing lists • Print Formatter • Menu-driven tutorial

VEDIT and CompuView are registered trademarks of CompuView Products, Inc. BRIEF is a trademark of UnderWare, Inc. PMATE is a trademark of Phoenix Technologies Ltd. Norton Editor is a trademark of Peter Norton Computing Inc.

# CompuView

1955 Pauline Blvd., Ann Arbor, MI 48103 (313) 996-1299, TELEX 701821

CIRCLE 122 ON READER SERVICE CARD



# DR. DOBB'S C TOOLBOX

## Dr. Dobb's Toolbook of C

**T**his authoritative reference contains over 700 pages, including the best C articles from Dr. Dobb's Journal along with new material. You'll find hundreds of pages of valuable C source code, including a complete compiler, an assembler, and text processing programs.

Toolbook of C Item #005 \$29.95

## Small C Handbook & Small C Compiler

**T**his compiler and handbook provides all you need for learning how compilers are constructed, and for learning C at its most fundamental level. You'll find a discussion of assembly language concepts and program translation tools, and of how to generate a new version of the compiler. Full source code is included. Please specify MS/PC-DOS or CP/M: Kaypro, Osborne, Apple, Zenith Z-100 DS/DD, 8" SS/SD.

CP/M Compiler & Handbook Item #006B \$37.90

MS/PC-DOS Compiler & Handbook Item #006C \$42.90

## Small Windows: A Windowing Library for Small C

**S**mall-*Windows* is a complete windowing library for Small-C. The package contains: 18 video functions written in assembly language; 7 menu functions that support both static and pop-up menus; and 41 window functions, including functions to clean, frame, move, hide, show, scroll, push, and pop windows. A file directory facility illustrates the use of the window menu functions and provides file selection, renaming and deletion capability. Two test programs are also included. For PC/MS-DOS systems only. Documentation and full source code is included.

Small-*Windows* Item #109 \$29.95

## Small-Tools: Programs for Text Processing

**T**his package of Small-C programs performs specific, modular operations on text files. It is supplied as source code with full documentation. With the Small-C Compiler, you can select and adapt these tools to meet your needs. Please specify MS/PC-DOS or CP/M: Kaypro, Osborne, Apple, Zenith Z-100 DS/DD, 8" SS/SD.

Small-Tools Item #010A \$29.95

## Small-Mac: An Assembler for Small-C

**T**his package includes: a simplified macro facility, C language expression operators, object file visibility, descriptive error messages and an externally defined instruction table. Source code and documentation is included. CP/M only. Please specify: Kaypro, Osborne, Apple, Zenith Z-100 DS/DD, 8" SS/SD.

Small-Mac Item #012A \$29.95

## Special Packages

CP/M C Package

Save Over \$27!

**R**ecieve: Dr. Dobb's Toolbook of C, The Small-C Handbook and Small-C Compiler, Small-Mac Assembler, and Small-Tools Text Processing Programs. Only \$99.95!

CP/M Package Item #005A \$99.95

MS/PC-DOS C Package

Save \$22!

**R**ecieve: Dr. Dobb's Toolbook of C, The Small-C Handbook and MS/PC-DOS Addendum, Small-C Compiler, Small-Tools Text Processing Programs and Small Windows. Only \$109.95!

MS/PC-DOS Package Item #005W \$109.95

C Disk Formats

Please indicate MS/PC-DOS or CP/M. For CP/M specify: Apple, Osborne, Kaypro, Zenith Z-100 DS/DD, 8" SS/SD.

**To Order:** Return this order form with your payment to M & T Books, 501 Galveston Dr., Redwood City, CA 94063.

Or, call TOLL-FREE 800-533-4372 Mon-Fri 8AM-5PM.  
(In CA call 800-356-2002)

## ORDER FORM

Name \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ Zip \_\_\_\_\_

Item #	Description	Price
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____

Subtotal \_\_\_\_\_

CA residents add sales tax \_\_\_\_\_ % \_\_\_\_\_

Add \$2.25 per item for shipping \_\_\_\_\_

TOTAL \_\_\_\_\_

☐ Check Enclosed. Make Payable to M & T Publishing Inc.

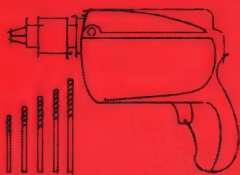
Charge my ☐ VISA ☐ M/C ☐ Amer. Exp.

Card # \_\_\_\_\_ Exp. Date \_\_\_\_\_

Signature \_\_\_\_\_

3128C





## POWER TOOLS for SYSTEM BUILDERS™

TSF is owned and operated by programmers, so we understand your needs. We believe and practice integrity in all business dealings. We advertise real prices and offer the best possible terms to all customers. We provide prompt delivery of current product versions.

We carry only products which we have personally used or which come with strong recommendations from developers we respect. We have the technical capability and the interest to search for and find new products to meet your specific needs. We will gladly provide quotes for volume purchases

or for products outside our normal product line. We consider it our job to help you get answers when publisher's technical support is unresponsive. At TSF service means more than a pushy sales pitch and a \$2 lower price.

We accept checks, Visa, Mastercard, American Express and COD. We charge your card only when we ship and do not add a surcharge. We provide free UPS delivery on software orders over \$100 (\$3 delivery charge on orders under \$100). Our COD fee is \$4. We allow return privileges on most products. Give us a try.

June Special: BDT only \$75 -- Save 25%

## Basic Development Tools™

**Powerful "Automatic Programming" Tools that  
Save You Hours of Valuable Time.**

The novice or power programmer can easily add these professional features.

- Screen Builder
- B+ Tree Disk File Access
- EZ Screen Pop-Up Windows
- Help screen message system

BASIC Development Tools™ (BDT™) is a set of four compatible tools which greatly increase the quality of your programs while requiring a minimum investment in money, code and debugging time.

### 1. Screen Builder System

Translates a painted screen image into BASIC code that you merge or include into your programs.

### 2. B+Tree Data Manager

Provides a very fast file access system that can be used for both sequential access and for random access with up to 40 keys. Complete source provided.

### 3. EZ Screen Pop-up Window Manager

Lets you easily insert and remove menus, windows and notepads from your program's display. Assembler routines (which you call from Basic) are used to automatically save and restore the display buffer, providing full screen management without slowing down execution speed.

### 4. Help Message System

Allows the creation of context sensitive help messages in your application program.

BDT gives you four great tools to improve the quality of your programs and to improve your own programming productivity. The BDT tools can be used separately or together and are compatible with BASICA, GWBASIC, Quick Basic™ and Borland Turbo Basic™. Includes two diskettes and a 220 page manual. No royalties. At a list price of \$99 its a bargain -- at TSF's June special price of \$75 its a steal. Don't delay, call or write now to place your order! 30 Day money back guarantee.



**STERLING CASTLE™**  
Sterling Castle, 702 Washington St., Suite 174,  
Marina del Rey, CA 90292



CIRCLE 230 ON READER SERVICE CARD

## The Software Family

649 Mission Street  
San Francisco, CA 94105

**(1)800-443-7176**

In California or outside U.S.

**(415)583-4166**

**Send Or Call For  
Comprehensive Free  
Catalog**

TSF carries a complete line of software and hardware tools for programmers and system engineers. We offer "big name" products at competitive prices and a large selection of hard-to-find products that save you hours of research, evaluation and development work. Our catalog includes hype-less product descriptions that explain what our products do to improve your productivity and your product's quality. Call or send the coupon for your free copy.

\_\_\_ Please send me a free catalog.  
I'm most interested in:  
\_\_\_ Basic \_\_\_ C \_\_\_ Pascal  
\_\_\_ Turbo Pascal \_\_\_ Other Languages  
\_\_\_ dBase \_\_\_ Technical Publishing

\_\_\_ Please send the following products:

Name: \_\_\_\_\_

Address: \_\_\_\_\_

City/State/Zip: \_\_\_\_\_

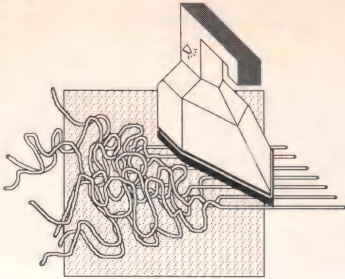
Credit Card: \_\_\_\_\_

Expiration Date: \_\_\_\_\_

Mail to: TSF, 649 Mission Street, San Francisco, CA 94105



# THE STATE OF BASIC



## BASIC Functions

In this issue we'll look at user-defined functions as implemented by QuickBASIC, Turbo BASIC, True BASIC, and Better BASIC.

QuickBASIC 2.0 permits user-defined nonrecursive functions to extend over multiple lines. All function names must start with the letters *FN*. A data type symbol may be required, depending on the data type returned by the function and any global default name declarations used. A function has an optional list of scalar arguments (no arrays are allowed) with all the arguments passed by value. Multiline functions end with an *END DEF* statement and can use *EXIT DEF* to exit from the function.

Most of the new BASIC dialects allow passing of all arguments by value. A function with no arguments that returns a simple value is one way of implementing Pascal-like constants; you cannot accidentally alter the function's value, as may be the case with a variable.

Nonparameter variables that are used within the functions are global. To avoid undesirable side effects and to localize such variables, include them in a *STATIC* declaration. This technique ensures that new addresses are assigned to these variables every time a *STATIC* declaration is encountered.

Turbo BASIC functions are similar to those of QuickBASIC, with the following differences:

- Turbo BASIC functions can be recursive.
- Local variables are declared inside functions using the *LOCAL* keyword.
- The *SHARED* keyword is used to explicitly declare global variables.

Turbo BASIC offers *LOCAL* and *STATIC* declarations, which give programmers the ability to explicitly declare local nonstatic variables. In Turbo BASIC, local arrays are first declared in the *LOCAL* list and then dynamically dimensioned using *DIM DYNAMIC*.

True BASIC implements user-defined functions in a slightly different way from that of the previous two BASIC dialects. First, function names do not have to start with the letters *FN*. The price to pay, however, is the use of *DECLARE DEF* declarations to inform True BASIC of the function

```
DEF FNFACT(N%)
' QuickBASIC nonrecursive factorial function
STATIC I%, F ' static local variables
F = 1 ' Initialize
' loop to get factorial
FOR I% = 2 TO N%
    F = F * I%
NEXT I%
FNFACT = F
END DEF
```

### Example 1: QuickBASIC listing for a nonrecursive factorial function

```
DEF FNFACT(N%)
' Turbo BASIC recursive factorial function
IF N% > 1 THEN
    FNFACT = N% * FNFACT(N%-1)
ELSE
    FNFACT = 1
END IF
END DEF
```

### Example 2: Turbo BASIC listing for a recursive factorial function

```
DEF Fact(N)
! True BASIC recursive factorial function
IF N > 1 THEN
    LET Fact = N * Fact(N-1)
ELSE
    LET Fact = 1
END IF
END DEF
```

### Example 3: True BASIC listing for a recursive factorial function

```
REAL FUNCTION: Fact
INTEGER ARG: N
EXTERNAL: Fact
10 IF N > 1 THEN RESULT = N * Fact(N-1) ELSE Fact = 1
END FUNCTION
```

### Example 4: Better BASIC listing for a recursive factorial function

```
REAL FUNCTION: LOGN
REAL ARG: X
REAL ARG: BAISE/OPT=10
10 RESULT = LOG(X) / LOG(BAISE)
END FUNCTION
```

### Example 5: Simple power function in Better BASIC that demonstrates the default parameter feature



names imported from external libraries and modules. True BASIC supports recursive multiline functions that take scalar- and array-type parameters. All the function parameters are passed by value.

In essence, True BASIC supports two levels of functions: internal and external. Internal functions are those located in the main program, before the unique *END* statement. All the variables in the internal function and not in the argument list are global, which enables internal functions to create and manipulate global variables. External functions are located either after the *END* statement of the main program or in an external library or module file. External functions defined in modules can access information through the argument list, *SHARED* variables, and *PUBLIC* variables. External functions in libraries have a strict data interface because they rely mainly on the argument lists. Argument lists in True BASIC can contain a file I/O channel number that allows file I/O, which provides another method of accessing large data that are stored in intermediate files.

Better BASIC approaches the implementation of functions in a Pascal-like fashion: the type of the function or its arguments is explicitly declared using data-type keywords and not symbols. Much of the discussion about Better BASIC procedures that appeared in the May column applies to functions. In addition:

- Function names do not need to start with the letters *FN*.
- The function type and name are declared on a separate line. This causes Better BASIC to respond interactively by creating a new workspace for the function and display and by displaying the memory available for it. Consequently, functions can use any range of valid line numbers without conflicting with other functions, procedures, and the main program.
- Parameters are declared by first listing their type, the keyword *ARG*., and the parameter name. Like Better BASIC procedures, parameters can be assigned default values.
- The result of the function is returned using the standard identifier

#### RESULT.

- Recursive functions need to declare the function and any local variables as external. This enables Better BASIC to allocate new addresses instead of using the same ones as in the calling function.

Examples 1–4, page 128, show versions of the factorial function written in each of the BASIC implementations discussed here. Recursive versions are used with all implementations

except QuickBASIC. Example 5, page 128, shows a simple Better BASIC function that returns the logarithm to any base. The default is base 10. That is, using the function *LOGN(X)* returns the base-10 logarithm, and *LOGN(X,BAISE)* returns the logarithm of *X* to base *BAISE*.

DDJ

Vote for your favorite feature/article.  
Circle Reader Service No. 12

## Lattice® Works

### LATTICE ANNOUNCES MICROSOFT WINDOWS SUPPORT IN VERSION 3.2

Version 3.2 of the Lattice MS-DOS C Compiler features full support for Microsoft Windows—including the “far,” “near,” and “pascal” keywords.

In addition, version 3.2 includes the ability to generate more than 64K bytes of static data and to declare objects larger than 64K bytes. It also includes improved support for ROM-based applications via the “const” data type. Version 3.2 is a significant release because it eliminates Microsoft’s claimed monopoly on future MS-DOS C development tools. Now that the Lattice MS-DOS C Compiler supports a window interface, programmers using Lattice C can avoid the problems caused by switching to a different compiler. \$500.00

### LATTICE NOW OFFERS ENHANCED AmigaDOS C COMPILER

Version 3.1 of the Lattice AmigaDOS C Compiler offers a new library with 100 more functions than the standard AmigaDOS C Compiler. What’s more, increased library modularity and new addressing modes help reduce load module sizes by more than 20%. The new version also features faster pointer and integer math, faster IEEE floating point routines, direct support of the

Amiga’s FFP format floating point library, and multi-tasking support.

With Version 3.1, Lattice has broken free of the reliance on the Amiga standard linker and object file format. This new release includes completely new expanded documentation, and a Lattice assembler and linker which remain compatible with previous software but allows professional programmers to take advantage of both the Amiga’s speed and the industry’s standardization.

Lattice AmigaDOS C Compiler with Lattice’s Text Management Utilities, \$225. Professional AmigaDOS C Compiler with, Text Management Utilities, Lattice Make Utility, Lattice Screen Editor, and the Metadigm MetaScope Debugger, \$375. AmigaDOS C Compiler \$150.

### LATTICE RELEASES NEW VERSIONS OF C CROSS COMPILER AND LINKER

Version 3.1 of the Lattice C Cross Compiler to MS-DOS and version 2.12 of the Plink86Plus Overlay Linker are now available for Sun and Apollo workstations as well as the DEC VAX Family of processors running VMS, UNIX or Berkeley UNIX.

All Lattice C Cross Compilers possess the same functionality and generate the same code as the native Lattice MS-DOS C Compiler. This allows users to take advantage of the larger systems’ speed and multi-user capabilities when creating applications for most popular PCs.

Contact Lattice Corporate Sales for details.



## Lattice

(800)533-3577 In Illinois (312) 858-7950 TELEX 532253 FAX (312) 858-8473  
INTERNATIONAL SALES OFFICES: Benelux: Ines Datacom (32)2-720-51-61  
Japan: Lifeboat, Inc. (03)293-4711 England: Roundhill (0672)54675  
France: Echosoft (1)4824.54.04 Germany: Pfotenhaur (49)7841/5058  
Hong Kong: Prima 85258442525 A.I. Soft Korea, Inc. (02)7836372  
Australia: FMS (03) 699-9899 Italy: Lifeboat Associates Italia (02) 46.46.01

CIRCLE 101 ON READER SERVICE CARD





## USING THE WRONG LANGUAGE CAN BE MURDER. SPEAK SMALLTALK/V



Let's talk languages. Programming languages like Turbo Pascal, C or Basic can be killers. To many, they're foreign, complex, and generally intimidating. Mistakes can be deadly.

With Smalltalk/V, you have an elegantly simple solution that puts the power and majesty of a major AI programming language on your PC or compatible. It makes no difference if you're an experienced programmer or just getting started. Smalltalk/V gives you an easy-to-use and flexible programming tool.

This is the same language used by leading software companies for their new product development. There are sound reasons for this. Smalltalk/V offers a totally integrated programming environment using the premier object-oriented language. You use natural language rather than complex programming codes. It puts Macintosh-type graphic features on a PC including overlapping windows, bit-mapping, pop-up menus, and a mouse interface. More than mere window dressing, Smalltalk/V delivers fully interactive windows that are easy to build and quick to modify.

But don't just take our word on it. Hear what the experts have to say:

*"This is the real thing folks. A super Smalltalk like this turns your PC into a hot workstation. It's fantastic... Highly recommended."*

John Dvorak  
Contributing Editor  
PC Magazine

*"The tutorial provides the best introduction to Smalltalk available."*

Dr. Andrew Bernat  
AI Expert Magazine

*"Smalltalk/V is the highest performance object-oriented programming system available for PCs."*

Dr. Piero Scaruffi  
Chief Scientist  
Olivetti Artificial  
Intelligence Center

Today, thousands of professionals, scientists and engineers are using Smalltalk/V to solve both simple and expert problems. Giving them a new dimension in computer applications for their PC.

Put new life into your PC by calling toll free 1-800-922-8255 and ordering Smalltalk/V today. Smalltalk/V by Digitalk, Inc., 9841 Airport Blvd., Los Angeles, CA 90045.  
(213) 645-1082.

**\$99.95**

Smalltalk/V comes with 10 starter applications including Prolog and each Application Pack adds several more. All source code is included. Supports 640 x 480 color graphics with color extension pack.

Smalltalk/V requires DOS and 512K RAM on IBM PC/AT/PS or compatibles and a CGA, EGA, Toshiba T3100, Hercules, or AT&T 6300 graphic controller. A Microsoft or compatible mouse is recommended. Not copy protected.

Turbo Pascal is a trademark of Borland International. IBM, IBM PC/AT/PS are trademarks of International Business Machines Corporation. Macintosh is a trademark of Apple Computer, Inc.

### TO ORDER CALL 1-800-922-8255 TODAY.

**60-DAY MONEY-BACK GUARANTEE\***  
Send check, money order, or credit card information to: Digitalk, Inc., 9841 Airport Boulevard, Los Angeles, CA 90045.

Credit Card ☐ VISA ☐ Mastercard

Card number: \_\_\_\_\_

Expiration date: \_\_\_\_\_

Name: \_\_\_\_\_

Street Address: \_\_\_\_\_

City/State/Zip: \_\_\_\_\_

Smalltalk/V	\$99.95
RS-232 Communications Application Pack	\$49.95
EGA/VGA Color Extension Pack	\$49.95
"Goodies" Application Pack	\$49.95

**SPECIAL OFFER:**  
Smalltalk/V and all 3 packs only **\$199.95**  
Shipping and handling (Outside North America) **\$15.00**

California residents add applicable sales tax  
**TOTAL \$ \_\_\_\_\_**

\*Unconditional 60-day money-back guarantee. Simply return to Digitalk, Inc. and your refund will be immediately forwarded to you.

**Smalltalk/V**  
digitalk inc.





## TALK OF THE TOWN

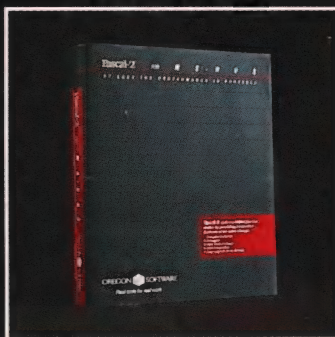
### One language supports this community.

That language is Pascal-2, now on the PC and producing the fastest, most compact code available. For the professional programmer, imagine what you can do with this power:

- Cut execution time by 20% to 200%
- Transport MS-DOS programs to VAX, PDP11, and 68000 machines with only minor adjustments
- Cut executable program size by up to 50%
- Use all of DOS-addressable memory through efficient large-memory model
- Speed error correction and save development turn-around time with sophisticated error checking and reporting
- Find and fix logical errors with the interactive source-level debugger
- Access DOS services

## Pascal-2<sup>™</sup>

FOR MS-DOS



and network files ■ Call Microsoft FORTRAN, C, Pascal, and assembler ■ Upgrade from TURBO Pascal with compatible strings, equivalent procedures and access to TURBO graphics.

### Plus!

- Intel CEL87 mathematical library for scientific computing
- A special interface between Pascal-2 and the programmable BRIEF text editor (editor optional).
- Certified ISO standard Level 1.

**Dramatically improve your productivity and introduce your PC software to the VAX next door.**

Call or write OREGON SOFTWARE, INC.  
6915 SW Macadam Avenue,  
Portland, OR 97219 (800) 367-2202  
TWX: 910-464-4779 FAX: (503) 245-8449

OREGON  SOFTWARE

*Real tools for real work*

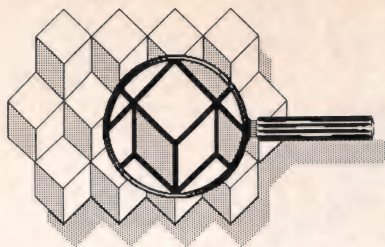
## AT LAST THE PERFORMANCE IS PORTABLE

The following are trademarks: Oregon Software, Pascal-2, Oregon Software, Inc.; IBM, PC-AT, PC-DOS International Business Machines Corporation; Intel, Intel Corporation; MS, Microsoft Corp.; TURBO Pascal, Borland International, Inc.; BRIEF, UnderWare Corp.; PDP, VAX, Digital Equipment Corp.

CIRCLE 357 ON READER SERVICE CARD



## OF INTEREST



## Languages

**Computer Crossware Labs** has introduced Real BASIC, a BASIC interpreter for the Atari 520 and 1040 ST that executes BASIC code 20 to 100 times faster than ST BASIC while maintaining full compatibility. It contains an in-line Motorola-compatible assembler that allows you to switch to assembly language without leaving the interpreted BASIC environment and to have full access to BASIC variables while in assembly mode. Real BASIC's features include an integrated, full-screen editor that supports ST BASIC commands and some Micro Emacs commands; extended graphics instructions; and function calls that deal with the mouse and joystick ports. Real BASIC sells for \$69.95. Reader Service No. 16.

Computer Crossware Labs Inc.  
516 Fifth Ave., Ste. 507  
New York, NY 10036  
(212) 677-3686

**Digitalk** has released two optional extension kits to accompany Release 1.2 of Smalltalk/V for PC-DOS machines. The first kit integrates full EGA color capabilities into the Smalltalk/V environment, greatly enhancing the system's bit-mapped graphics. The second kit, called Goodies, offers several new programming kernels and lets you extend Smalltalk to handle applications that require discrete event simulation, forward-chaining interface operations, and connection to external sensors and instrumentation. Release 1.2 of Smalltalk/V is priced at \$99, and updates to 1.1 can be downloaded from CompuServe and BIX or obtained on disk from the company for \$10. The optional extension kits cost \$49 each. Reader Service No. 17.

Digitalk Inc.  
5200 W. Century Blvd.  
Los Angeles, CA 90045  
(213) 645-1082

**Rational Visions** is shipping a full-featured PROLOG programming system for the Atari ST. The system uses the Edinburgh standard syntax, making it compatible with most popular tutorials on PROLOG. Features include an Emacs-style text editor within the system's interpreter that allows you to interact with external disk files, the internal knowledge base, or invoke PROLOG goals from within the editor; a built-in grammar-rule translator that allows the development of natural-language interfaces; support for floating-point values; and extended math functions. Also, all PROLOG's metaprogramming primitives have been implemented, as have Atari-specific features such as GEMDOS primitives and interfaces to VDI and AES. The system is not copy-protected and supports user-written applications free of licensing restrictions. The package sells for \$39.95. Reader Service No. 18.

Rational Visions  
7111 W. Indian School Rd., Ste. 131  
Phoenix, AZ 85031  
(602) 846-0371

**Datalight** has introduced Optimum-C, a full-featured global optimizing C compiler for IBM PCs and compatibles that supports the Unix System V C language along with several proposed ANSI extensions. Features include 8087 and software floating-point support, strong type checking, ROMable code generation, a MAKE program, MS-DOS object files format, Lattice C compatibility, and full library/start-up source code. Support is provided for compact-, small-, medium-, and large-memory models. Source licenses and support contracts are available for the compiler. Optimum-C sells for \$139. Reader Service No. 19.

Datalight  
P.O. Box 82441  
Kenmore, WA 98028  
(206) 367-1803

## Tools

Devpac Amiga, from **HiSoft**, is a

68000 program development system for the Amiga computer that features a combined editor and assembler and a symbolic disassembler/debugger. The full-screen editor can be used from either the mouse or from the keyboard. The assembler is a complete, fast, macro assembler that supports include files read from disk, conditional assembly, and Motorola-standard macro handling and that can produce executable or linkable code. The debugger includes all the expected commands, such as breakpoints and single-stepping, and also allows you to use your original symbols when debugging programs. It uses its own screen for its display so as not to disturb that of other programs. Devpac Amiga is available in the United States from Apex Resources. It runs on any Amiga with at least 512K RAM and costs \$99.95. Reader Service No. 20.

Apex Resources  
129 Sherman St.  
Cambridge, MA 02140  
(800) 343-7535  
In MA (617) 876-2505

**Numerical Algorithms Group (NAG)** now offers 172 selected routines from its extensive NAG Fortran Library for use on workstations and personal computers. The NAG Fortran Workstation Library provides the routines most frequently used by engineers and PC programmers. Each routine includes an example program, the source, and data results to illustrate the routine's usage. The NAG Fortran Workstation Library is available for the DEC MicroVAX, IBM PC line, and Sun workstations. License fees range from \$1,296 for a single workstation to \$384 each for 11 or more. Reader Service No. 21.

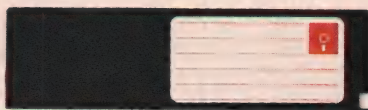
Numerical Algorithms Group Inc.  
1101 31st St., Ste. 100  
Downers Grove, IL 60515  
(312) 971-2337

P-tral, from **Woodchuck Industries**, is a native code BASIC-to-Pascal translation package that translates IBM BASICA/MS BASIC to Turbo Pascal. The program is interactive and lets you pick out or name subroutines as well as rename variables not fitting Pascal criteria. The program works



# Think fast! Pick the better fit...

Our 5th Year Bonus!  
Mention this ad when ordering  
and get your choice of a V-20-8  
(replaces 8088) or \$20 off on  
your Battery Backup purchase!



## FLOPPY DISK.

- Fills time between coffee breaks
- Makes a hard disk seem *fast*
- Your computer appears busy (even if you aren't!)
- Wears out moving parts

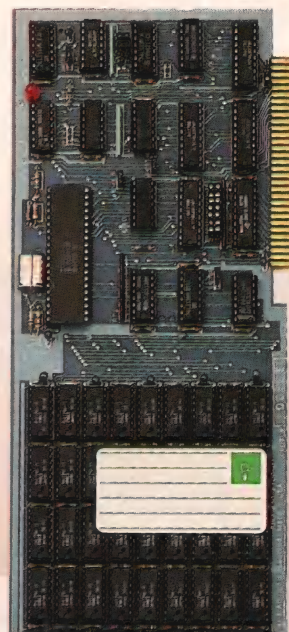
...for YOUR demanding tasks.

**SURPRISE!** Neither is memory mapped, so they don't affect your precious Main Memory. Both retain data indefinitely - even with the computer turned off.

**THE SEMIDISK SOLUTION.** You could invest in a series of "upgrades" that turn out to be expensive band-aids without solving your real problem. Even those "Accelerator" and "Turbo" boards do little to speed up disk-bound computers. If your applications spend too much time reading and writing to disk (and whose don't?), you won't want to settle for anything less than a SemiDisk disk emulator. The SemiDisk comes in 512K and 2Mb capacity. More boards may be added to make up to an 8 Megabyte SemiDrive!

**SPEED THAT'S COMPATIBLE.** PC, XT or AT, if you need speed, the SemiDisk has it. How fast? Recent benchmarks show the SemiDisk is from 2 to 5 times faster than hard disks, and from 25% faster (writing) to several times faster (random reads) than VDISK and other RAMdisk software that gobble up your main memory.

**MEMORY THAT'S STORAGE.** Using our small external power supply, with battery backup, your data remains intact through your longest vacation or even a seven-hour power failure!



## SEMIDISK Disk Emulator.

- Gets that job done **NOW**
- Makes a hard disk seem *slow*
- Maximizes your productivity with anything from databases to compilers
- Totally silent operation

**CELEBRATE WITH US!** Now, SemiDisk celebrates its fifth birthday with a special offer for IBM-PC owners. Buy a SemiDisk now and we'll include an 8 MHz V-20 micro-processor (replaces the 8088) to make your new SemiDisk run even faster. Don't need the V-20? We'll take \$20 off the price of your Battery Backup Unit!

	512K	2Mbyte
IBM, PC, XT, AT	\$495	\$ 795
Epson QX-10	\$495	\$ 995
S-100 SemiDisk II	\$795	\$1295
S-100 SemiDisk I	\$299	-----
TRS-80 II, 12, 16	\$495	\$ 995
Battery Backup	\$130	\$ 130

**Someday you'll get a SemiDisk.  
Until then, you'll just have to...wait.**

# SemiDisk



SemiDisk Systems, Inc., 11080 S.W. Allen Blvd., Beaverton, Oregon 97005 (503) 626-3104

CIRCLE 85 ON READER SERVICE CARD



best with a hard disk and requires MS-DOS or PC-DOS 2.1 or later with ANSI.SYS. P-tral sells for \$179. Reader Service No. 22.

Woodchuck Industries Inc.  
340 W. 17th St., #2B  
New York, NY 10011  
(212) 924-0576

RTC Plus, a FORTRAN- and RATFOR-to-C translator package from **Cobalt Blue**, allows you to tap old FORTRAN code for new C development. The translator is designed for translating non-I/O FORTRAN libraries and code in which I/O is concentrated in a few routines. Also, more than 95 percent of STUG's RATFOR statements are supported by RTC Plus for complete translations of clean RATFOR code. RTC Plus runs under MS-DOS, Version 2.2 or later and costs \$325. Reader Service No. 23.

Cobalt Blue  
1683 Milroy, Ste. 101  
San Jose, CA 95124  
(408) 723-0474

**TurboPower Software** has released an upgrade to T-DebugPLUS, a symbolic run-time debugger for Turbo Pascal. The new version (1.04) allows Turbo Pascal programmers to debug code in overlays and access CPU registers and memory. Other improvements include easier customization of the debugger and a new MAKELST utility program that creates a commented disassembly listing of Turbo Pascal programs. T-DebugPLUS requires Turbo Pascal 3.0 and a PC-DOS machine with 256K RAM. It sells for \$60 (\$10 for upgrade). Reader Service No. 24.

TurboPower Software  
3109 Scotts Valley Dr., Ste. 122  
Scotts Valley, CA 95066  
(408) 438-8608

**Laney Systems** has released StruBAS 2.0, a structured BASIC toolkit for application development with QuickBASIC and IBM BASIC 2.0 compilers in network and single-user environments. Integrated tools include enhanced structured programming facilities, screen handling, a Btrieve interface, StruBAS/ISAM for those not wishing to buy Btrieve, an object library of utility subroutines, and utility programs. A preprocessor ex-

pands BASIC with new commands to produce a more readable and structured language while making all BASIC's features and constructs available as well. StruBAS programs are translated by the preprocessor to compilable BASIC, which is then compiled and linked to produce executable programs. StruBAS is not copy-protected, and no royalties are charged for programs written using the package. It requires a PC-DOS machine with 320K RAM and a hard disk and sells for \$495. Reader Service No. 25.

Laney Systems Inc.  
3 Office Park Dr., Ste. 105  
Little Rock, AR 72211  
(501) 225-7755

BASIC Program Analyzer, from the Expert Systems Division of **Expert Information Systems**, is a rule-based system that reads Microsoft's GW-BASIC program files in ASCII format and performs analysis, listing, and cross-reference functions upon them. The analyzer prints individually tabulated cross-references of line numbers, variables, keywords, quoted strings, and numerical constants found in theograms. Error, warning, and comment messages are inserted where applicable into tabulated cross-references, and an ASCII file is produced of all portions of the printout, including the cross-references. Optional parameters can be used to modify some of the analyzer's operations, such as testing for compiler syntax compatibility and for redirecting output. BASIC Program Analyzer is available for most MS-DOS computers and costs \$99.95. Reader Service No. 26.

Expert Information Systems Inc.  
Expert Systems Division  
P.O. Box 1310  
El Campo, TX 77437-1310  
(409) 543-9222

### Libraries

**United States Software Corp.** has released a 68HC11 floating-point library. This complete math package operates on the 68HC11, is designed in a modular format, and is delivered in source assembly language for maximum flexibility. The library conforms to the IEEE 754 Floating Point Standard. The 68HC11 FPAC/

DPAC is available in either single-precision or double-precision formats. The library includes trigonometric, logarithmic, and exponentiation functions, and data-conversion and floating-point utility procedures. It is available on IBM PC-compatible media for a one-time fee of \$950. Reader Service No. 27.

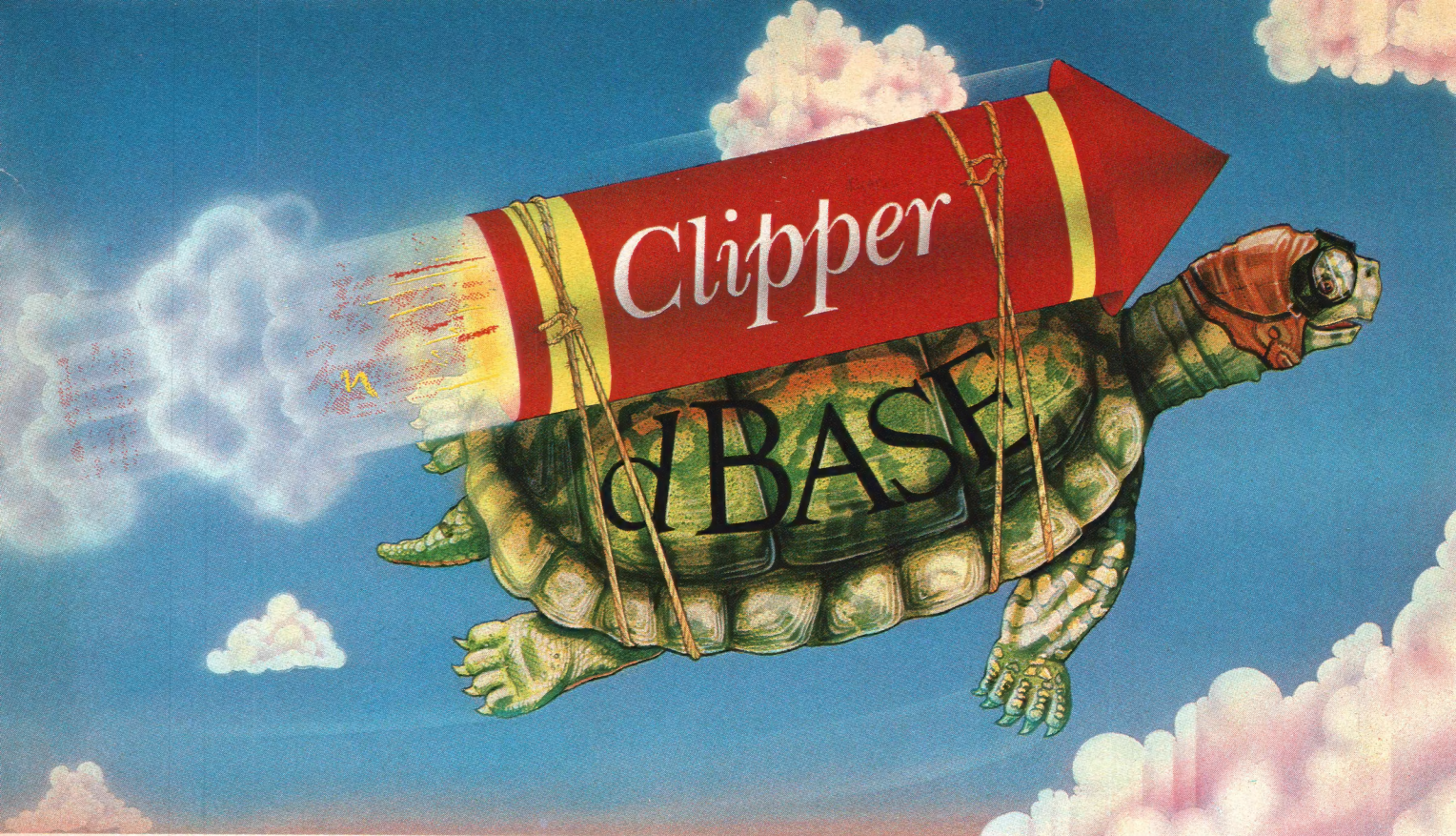
United States Software Corp.  
14215 N.W. Science Park Dr.  
Portland, OR 97229  
(503) 641-8446

LINLIB, from **Information and Graphic Systems**, is a library of C routines for developers of scientific and statistical software. The package performs all basic operations on vectors and matrices, with no limitation on size of vector or matrix. Of major importance are the routines that factor matrices (LU, QR, and Cholesky factors). These factorizations enable you to solve any kind of system equations—underdetermined, determined, and overdetermined. Also included in LINLIB are routines based on the B-spline for manipulating splines. LINLIB sells for \$150. Reader Service No. 28.

Information and Graphic Systems  
15 Normandy Ct.  
Atlanta, GA 30324  
(404) 231-9582

DDJ





Real programmers don't use dBASE. Or do they?

We're finding that some very swift programmers are using it to write some very fast applications, and are completing their projects much more quickly.

But they cheat.

They use our Clipper™ compiler to combine dBASE™ with C and assembler.

With dBASE used like pseudo-code, they can then quickly create prototypes that actually run.

Then, with dBASE doing the high-level database functions, they use our Clipper compiler to link in C or assembly language modules from their own bag of tricks.

And they're finding that they're linking in less than they expected because Clipper compiled code runs so fast and because of Clipper's built-in enhancements.

Clipper includes easy networking that provides file and record locking the way it should be done.

Fast screens that can be treated as memory variables and eliminate the need for direct screen writes and all that tortuous heap management code.

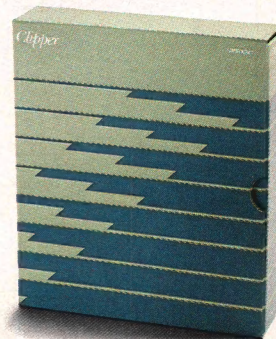
Box commands that make windowing a breeze. And more.


So if you'd like to use your time more productively, check us out:

Nantucket Corporation,  
12555 W. Jefferson Boulevard,  
Los Angeles, CA 90066.

Or if you're on deadline, call  
(213) 390-7923 today.

Clipper could get you out of the soup.



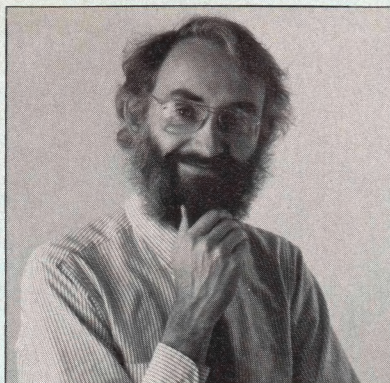
 **nantucket®**

© Nantucket Corporation 1987. Clipper is a trademark of Nantucket Corporation; dBASE isn't. In Europe: Nantucket Corporation (Europe) 2 Bluecoats Avenue, Fore Street, Hertford, Herts SG14 1PB Telephone 0992 554621.

**CIRCLE 220 ON READER SERVICE CARD**



## SWAINE'S FLAMES



I think I may be the first person who has ever actually seen a computer bug.

It came to me at the end of a long and disillusioning night when my body, no doubt in sympathy with the new punitive tariffs on Japanese goods, began rejecting a dinner of overripe sushi. En route from bathroom to loft, I heard an odd tapping sound from the office and stopped to investigate.

In my collywobbled state I had neglected to turn off the Macintosh, and what I saw in the glow from the screen was a bug—it looked a lot like a cockroach—repeatedly climbing to the top of the Mac and leaping to the keyboard to enter a character. The exertion must have been tremendous, and he looked pretty beat when he finally finished and crawled into the disk drive. I'm not surprised he didn't bother with capitalization or punctuation (the bottom row of keys was almost too long a jump for him).

He left this behind.

boss i m afraid that the computer curmudgeon john dvorak has been talking about our beloved dr dobb s journal of whatever again if dr. dobb s became outspoken it would be dangerous dvorak said in one of those tasty magazines you leave lying around this office i wish you would leave a crust of bread i m getting tired of paper and glue

i wonder does he mean like that 1940s science fiction writer was dangerous when he explained how an a bomb would work or like the white house press corps would be dangerous if it decided it wasn t the white house s press corps

or like sesame street would be dangerous if it taught children to question their teachers and their parents or maybe like infoworld was dangerous when it was publishing dvorak

sometimes i think that john dvorak is a modelless dialog box the max headroom of the personal computer industry

how could dr dobb s become dangerous is what i d like to know maybe by examining legislator s voting records on software issues or by doing a study of compiler pricing vs value or by publishing detailed specs on data formats of commercial products so that data held hostage by obsolete applications could be liberated or by showing how to develop software for the 386 without waiting for you know who or by questioning the need for an operating system on a personal computer at all

i used your modem to call dvorak s number and talked with his dog sparky maybe dr dobb s should list the companies that preannounce products to outflank their competitors sparky suggested or do their beta testing in

release version one point oh or cripple their products with copy protection tricks or refuse to eliminate known bugs what s this about bugs i asked him but he just barked

one thing i know is that dr dobb s could stand to be a little more rough and tumble a bit more capricious and corybantic there s a dance in the old dame yet and if that s dangerous it s fine by me please save this file as i can t work the mouse nor would i want to i haven t always had the best of relations with mice yours archy

Scarab journalism was invented by Don Marquis, a *New York Sun* columnist, when he found a cockroach composing in his office late one night some seventy years ago, learned that the bug was a reincarnated vers libre poet, and cannily conned the poet roach into meeting copy deadlines for him for the next ten years.

The saga of archie the cockroach itself allegedly owes its existence to the newspaper account of a type-writing rat who plied his or her art by night circa 1916 in a Dobb's Ferry, New York, garage. By some odd symmetry in the transmigration of souls, a Dobb has once again played host to verse vermin.

Fine by me.

*Michael Swaine*

Michael Swaine  
editor-in-chief





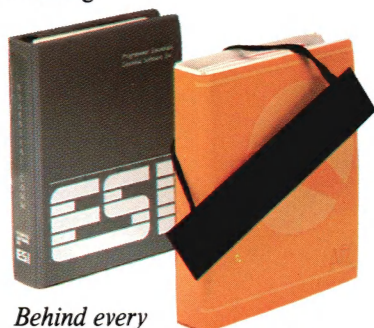
**TURBO C VERSION**  
Now Available  
call for details

## You'd Be Surprised Who Uses Essential Graphics Functions

And some of our well-known customers would be just as surprised if they saw themselves in this ad. We don't splash their names all over the place as a matter of professional courtesy.

Let's face it. Just because someone else uses a product is not reason enough to buy it. The clincher is that our programs run a *documented 40% faster* than the closest competitor. To complete the picture, our code is up to *75% smaller* due to efficient coding and the granularity of functions.

However, should curiosity get the best of you, call us at 201-762-6965 and we'll drop a few highly impressive names on you.



*Behind every great program is a great library.*

### Draw Your Own Conclusions

When you're responsible for a project that includes advanced graphics, "graphics windowing," or character font manipulation, Essential Graphics is the clear choice. We've taken the grind out of graphics programming and replaced it with speed and versatility.

### No Royalties, 30 Day Guarantee

We believe that selling you a programming tool does not make us your co-authors. So we don't charge any royalties or run time fees. If within 30 days you don't find our library satisfactory, dump the whole thing and receive a complete refund.

### Functions At A Glance

#### Features:

- Fastest functions available
- Dots, Lines, Circles, Arcs, Pies, Bars
- Manipulate character fonts
- Move blocks, do animation
- User definable patterns
- Seed filling in a boundary
- Clipping on screen coordinates

#### Devices Supported:

- IBM, Epson, Oki printers
- HP Plotters, HP Laser Jet
- Microsoft, Logitech Mice

#### Graphics Adapters:

- IBM Color Graphics
- IBM Enhanced Graphics
- Hercules Graphics
- AT&T, Olivetti Graphics
- Tecmar Graphics Master
- Others (Call)

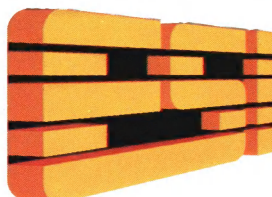
#### Compiler Compatibility:

- Microsoft, C, Fortran, Pascal
- Lattice C, Aztec C
- Computer Innovations C86, DeSmet C
- Wizard C, Mark Williams

**\$250.00**

### Do Your Homework

The library you buy can influence the rest of your programming life. We encourage you to do some checking before making a decision. When you've done your homework, you'll choose Essential. Call our support staff of experienced C programmers and find out before you buy how things will be after your check clears.



**To order or for support  
call: 201-762-6965**

### For foreign orders contact:

England: Gray Matter Tel. (0364) 53499  
Japan: Lifeboat Inc. of Japan Tel: 293 4711  
West Germany: Omnitex Tel. 07623-61820

**Essential Software, Inc.**

P.O. Box 1003, Maplewood, New Jersey 07040

**CIRCLE 138 ON READER SERVICE CARD**



# HAUPPAUGE

## Thrust your math into high gear!

**H**AUPPAUGE has a hunch you need faster math calculations. So we've launched our 287 FaST10—the highest speed math coprocessor available for PC/AT's and all AT compatibles.

Now you can speed up your math calculations without worrying about data safety. Our FaST10 has a self contained high speed clock that speeds up math coprocessor operations without jeopardizing accuracy.

The FaST10 accelerates programs including 1-2-3, Framework, AutoCAD and Symphony (to name just a few)—by as much as 250%!

Our FaST10 is compatible with: 6 or 8 MHz IBM PC/AT's, the Compaq DeskPRO 386 and all 286-based PC's like the Compaq DeskPRO 286 & Portable 286.

### We have all the Coprocessors you need!

FaST modules, which operate at full speed, for the PC/AT, Compaq DeskPRO 386 and PC/AT compatibles:

287 FaST 10—full speed 10MHz .....	\$469
287 FaST 8—full speed 8 MHz .....	\$379
287 FaST 6—full speed 6 MHz .....	\$249

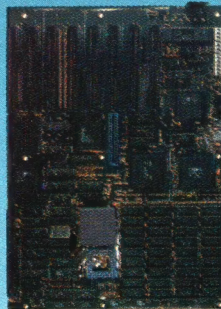
Math coprocessors for AT compatibles and accelerator cards:

287-10 Chip .....	\$439
287-8 Chip .....	\$349
287-6 Chip .....	\$219

Math coprocessors for PC/XT's & higher speed compatibles:

87 Chip—4.77 MHz ...	\$129
87-2 Chip—8 MHz .....	\$195

Save your PC from early retirement. Get Hauppauge's new **386 MotherBoard!** Hauppauge has a smart way to give your IBM PC or XT the speed and compatibility of the 80386.



It's our 386 MotherBoard. With a 32-bit, 16 MHz 80386 processor and 1 Mbyte of high speed memory, it scores high on desktop publishing, CAD graphics, engineering and business applications. 386 MotherBoard is a motherboard replacement that's 100% form fitting. So installation takes just 20 minutes.

To accept new generations of DOS, we gave it PC/AT BIOS, I/O compatibility and up to 16 Mbytes of high speed expansion RAM.

And our 386 MotherBoard has a lot more on the ball than simple accelerator cards. Because in addition to speed, it has a 16 bit expansion slot. Plus a 32 bit slot for high speed memory expansion. There's also a 387 math coprocessor socket for the ultimate in high speed math calculations, and a clock/calendar with battery backup for true PC/AT compatibility.

386 MotherBoard also comes with very intelligent prices:

386 MotherBoard/PC .....	\$1495
386 MotherBoard/XT .....	\$1495
387 Math coprocessor .....	\$695

## Hauppauge

Your math coprocessor specialists

**Hauppauge Computer Works, Inc.**  
358 Veterans Memorial Highway,  
Commack, N.Y. 11725  
Phone: 516-360-3827